COMILLAS
UNIVERSIDAD PONTIFICIA
ICAI

MASTER'S DEGREE IN INDUSTRIAL ENGINEERING

FINAL MASTER THESIS

Virtual Data Concentrator: Testing, validation, and documentation of containerization/virtualization architecture for tele management Data Concentrator application in distributed environments

Author: Emilio Juan Valdivielso Suárez

Director: Aurelio Sánchez Paniagua

Co-Director: Javier Matanza Domingo

Madrid

August de 2025

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

Virtual Data Concentrator: Testing, validation, and documentation of containerization/virtualization architecture for tele management Data Concentrator application in distributed environments

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el

curso académico 2025 es de mi autoría, original e inédito y

no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido

tomada de otros documentos está debidamente referenciada.

Fdo.: Emilio Juan Valdivielso Suárez Fecha: …15…/ …08…/ …2025…

[illegible]

Autorizada la entrega del proyecto

LOS DIRECTORES DEL PROYECTO

Fdo.: Javier Matanza Domingo Fecha: …15…/ …08…/ …2025…

Firmado por MATANZA DOMINGO JAVIER - ***4397** el día 14/08/2025 con un certificado emitido por AC FNMT Usuarios

Fdo.: Aurelio Sánchez Paniagua Fecha: …15…/ …08…/ …2025…

[illegible]

COMILLAS
UNIVERSIDAD PONTIFICIA
ICAI

MASTER’S DEGREE IN INDUSTRIAL ENGINEERING

FINAL MASTER THESIS

Virtual Data Concentrator: Testing, validation, and documentation of containerization/virtualization architecture for tele management Data Concentrator application in distributed environments

Author: Emilio Juan Valdivielso Suárez
Director: Aurelio Sánchez Paniagua
Co-Director: Javier Matanza Domingo

Madrid
August de 2025

Acknowledgements

I would like to express my gratitude to all those who have supported me both academically and personally in reaching this point in my career.

First, I would like to thank Iberdrola, in collaboration with the Universidad Pontificia de Comillas, for giving me the opportunity to carry out my master's thesis with access to the latest technology used in the Spanish distribution network, as well as the guidance of experts with extensive experience in the sector.

In the academic field, I would especially like to thank Aurelio Sánchez Paniagua for his constant support throughout the project, his patience in answering countless questions, and for contributing to my academic and professional development, always with a smile. I also want to thank Javier Matanza Domingo for his academic guidance and advice during the project, as well as during my time in the master's program as a professor. In addition, I would like to thank Francisco Fuentes Cervera for his help and technical explanations, and for guiding me through the laboratory testing phase.

On a personal level, I would like to thank and dedicate this project, which marks the end of my academic journey, to all those who have contributed to my growth as a person and as a student, and whose teachings have allowed me to close this chapter feeling proud of all the effort invested.

First and foremost, I want to thank my grandparents and my parents for their unconditional love and support, and for raising me with values rooted in effort and dedication. I am also deeply grateful to my parents for the tremendous financial effort involved in supporting my studies and this master's program. Finally, I want to thank my entire family and all my loved ones, especially those to whom I have not been able to dedicate as much time as I would have liked during these six years of study such as my partner, family, and friends.

In summary, thank you all for guiding me and being by my side during these 6 years.

CONCENTRADOR VIRTUAL DE DATOS: PRUEBAS, VALIDACIÓN Y DOCUMENTACIÓN DE LA ARQUITECTURA DE CONTENERIZACIÓN/VIRTUALIZACIÓN PARA LA APLICACIÓN DE CONCENTRADOR DE DATOS DE TELEGESTIÓN EN ENTORNOS DISTRIBUIDOS

Autor: Valdivielso Suárez, Emilio Juan.
Director: Sánchez Paniagua, Aurelio.
Entidad Colaboradora: Iberdrola i-DE

RESUMEN DEL PROYECTO

Este proyecto tiene como objetivo validar técnica y funcionalmente la virtualización del Concentrador de Datos (VDC) en entornos distribuidos de redes eléctricas, como alternativa al hardware embebido tradicional. Se ha realizado un análisis comparativo entre la plataforma previamente testada de Minsait y la solución de edge computing de Barbara, actualmente en fase de evaluación en Iberdrola i-DE, así como frente al concentrador embebido convencional. El estudio ha evaluado su capacidad para soportar aplicaciones críticas en centros de transformación. Se ha llevado a cabo el despliegue completo de un nodo edge basado en Barbara sobre hardware industrial. La equivalencia funcional con la solución actual embebida se ha demostrado mediante pruebas de configuración, interconectividad, adquisición de datos y ejecución de tareas programadas. Además, se ha desarrollado e implementado un caso de uso de regulación de tensión mediante cambiadores de tomas en carga (OLTC) directamente sobre el nodo virtualizado, evidenciando la capacidad de la plataforma para habilitar nuevas funcionalidades de automatización de red. Las pruebas de escalabilidad han confirmado que el VDC puede gestionar más de 7.000 contadores sin comprometer la integridad operativa. Un análisis económico adicional ha puesto de relieve la viabilidad del enfoque virtualizado bajo determinados marcos regulatorios. Más allá de la validación técnica, este trabajo contribuye a la Alianza E4S proponiendo definiciones de interfaces que mejoran la interoperabilidad entre componentes de edge computing. Los resultados sientan las bases para arquitecturas modulares, escalables e interoperables en futuros despliegues de redes inteligentes.

Palabras clave: Plataformas de Procesamiento Distribuido, Edge Computing, Nodo Edge, Centro de Transformación Inteligente, Interoperabilidad, Redes Inteligentes, Concentrador de Datos Virtual.

1. Introducción

La red de distribución eléctrica está experimentando un profundo proceso de digitalización, motivado por la necesidad de mejorar la eficiencia, la integración de energías renovables distribuidas (DERs) y los objetivos de descarbonización. Según Iberdrola, las redes eléctricas inteligentes son redes capaces de integrar de forma inteligente y dinámica las acciones de todos los usuarios conectados, ya sean generadores, consumidores o ambos, para suministrar electricidad de manera eficiente, sostenible, económica y segura [1]. La evolución hacia este modelo en las redes de distribución busca optimizar la fiabilidad y eficiencia mediante monitorización en tiempo real, flujos bidireccionales de energía y control adaptativo. En este contexto, la modernización de los Centros de Transformación (CT) juega un papel crucial, pues son nodos clave que conectan la red de media tensión con la de baja tensión y pueden aportar nuevas funcionalidades como la integración de DERs, el mantenimiento predictivo, la gestión dinámica de la carga y el procesado de datos de medida sin necesidad de enviarlos a la nube.

Actualmente, muchas de estas funciones en los CT dependen de equipos hardware propietarios como Concentradores de Datos (CD) de telegestión y Unidades Terminales Remotas (RTUs) dedicadas a supervisión y control remoto. Este enfoque basado en dispositivos específicos conlleva limitaciones importantes como la dificultad para encontrar proveedores alternativos (hardware y software altamente personalizados), ciclos de desarrollo y despliegue muy rígidos (incorporar nuevas funcionalidades puede llevar varios años), y falta de flexibilidad para adaptarse a cambios regulatorios o tecnológicos. Si bien la solución actual de la distribuidora Iberdrola i-DE ha logrado desplegar a gran escala, gestionando más de 120 000 posiciones de transformación y aproximadamente 11,6 millones de contadores, no ofrece suficiente adaptabilidad para nuevas exigencias de innovación o normativas. Esto se debe al carácter monolítico de los sistemas existentes y a la necesidad de actualizar miles de equipos físicamente para implementar cualquier mejora.

En respuesta, la tendencia actual apunta a utilizar plataformas unificadas tipo IED (Intelligent Electronic Device) en hardware genérico dentro del CT, integrando las funciones de concentrador, RTU y supervisor de baja tensión mediante virtualización o contenedorización. Aunque estándares como IEC 61850 han mejorado la interoperabilidad en subestaciones inteligentes, no resuelven por completo las limitaciones citadas [2]. Por ello, la combinación de la virtualización y el edge computing surge como la principal solución a este problema. El edge computing implica descentralizar el procesamiento de datos hacia nodos locales, reduciendo la latencia y permitiendo respuestas en tiempo real para aplicaciones como balanceo de carga, detección de fallos o análisis inmediato de datos, facilitando así la operación de la red BT. Por otra parte, la virtualización desacopla el software del hardware, haciendo posible que soluciones embebidas tradicionales se ejecuten en dispositivos genéricos. Esto permite unificar en un solo nodo funciones antes repartidas en varios equipos, reduciendo costes de capital y mejorando la flexibilidad operativa. Un caso de uso inicial clave es el concentrador de datos virtual (VDC), dada su importancia para recopilar y proveer datos utilizados por otras aplicaciones. Al virtualizar el concentrador, se puede adaptar más ágilmente a la evolución de protocolos y agregar nuevas funcionalidades o requisitos regulatorios sin reemplazar físicamente miles de dispositivos.

Para habilitar esta visión han surgido diversas plataformas de orquestación de aplicaciones en el edge. Soluciones industriales como Barbara IoT [3] u Onesait Phygital Edge de Minsait, entre otras plataformas de procesamiento distribuido mencionadas en [4], permiten el despliegue seguro y escalable de software en nodos edge heterogéneos. Estas plataformas proporcionan mecanismos de gestión centralizada de nodos y contenedores, comunicaciones seguras, y despliegue remoto de aplicaciones. En línea con estos avances, la plataforma tecnológica española Futured (que agrupa a Iberdrola y otras empresas eléctricas) propone transformar el CT tradicional en una Subestación Secundaria Inteligente y Automatizada con Infraestructura de Computación y

Telecomunicaciones (SASCTI) [5]. Paralelamente, la alianza E4S (Edge for Smart Secondary Substations) tiene como objetivo definir estándares y arquitecturas comunes que faciliten la colaboración entre distribuidoras (DSOs) y fabricantes, fomentando soluciones modulares y agnósticas al proveedor en lugar de sistemas monolíticos [6].

En este contexto de modernización, la distribuidora Iberdrola i-DE ha lanzado el proyecto PRADA, que durante el próximo periodo regulatorio (2026–2031) busca actualizar su infraestructura de medida, incluyendo migración de PRIME 1.3.6 a PRIME 1.4 y renovación de contadores antiguos a la versión SM 2.0. Estos cambios ofrecen la oportunidad de modernizar también los concentradores de datos en los CT, reemplazando equipos obsoletos (como concentradores con módem 3G integrado) por soluciones más avanzadas. En concreto, el proyecto propone comparar dos escenarios. El primero, consiste en despliegue tradicional de nuevos concentradores físicos (denominados TGUC) con hardware dedicado (incluyendo nodo base PRIME y supervisor BT integrados) en cada posición de transformador. Y el segundo escenario consiste en instalar nodos edge en los CT ejecutando aplicaciones virtualizadas, siendo el VDC la primera de ellas. Bajo este segundo enfoque, un único nodo industrial por subestación, conectado a módulos externos (nodo base PRIME y supervisor BT), podría reemplazar a múltiples concentradores propietarios. Iberdrola i-DE pretende investigar y validar la viabilidad técnica de esta virtualización del concentrador, esperando con ello mejorar la flexibilidad, interoperabilidad y escalabilidad de la red, así como evaluar su viabilidad económica dentro del modelo regulatorio español vigente

2. Definición del proyecto

El objetivo principal de este proyecto es validar técnicamente un Concentrador de Datos Virtual (VDC) implementado en una plataforma edge, y proponer una arquitectura modular que facilite su integración en entornos distribuidos y multi-vendor alineados con la iniciativa E4S. Para alcanzar este objetivo general, se plantearon varios subobjetivos específicos:

1. Análisis de plataforma edge: Estudiar críticamente la plataforma Barbara IoT en laboratorio, evaluando su arquitectura, capacidad para orquestar aplicaciones en nodos distribuidos y rendimiento bajo condiciones operativas reales. Se realizó también una comparación técnica con otra plataforma de edge computing (Onesait de Minsait) para identificar fortalezas, limitaciones y adecuación a requisitos de smart grids.

2. Documentación de despliegue: Documentar el proceso completo de implantación del VDC en entorno OT, incluyendo la instalación del sistema operativo Barbara OS en hardware industrial genérico y la configuración del nodo edge. Esto abarca la adaptación de la plataforma Barbara a la infraestructura de i-DE y la conexión con los dispositivos de campo.

3. Arquitectura modular e interoperable: Definir los requisitos y arquitectura que una plataforma de procesamiento distribuido (DPP) debe cumplir para ser agnóstica al proveedor, con énfasis en interoperabilidad, migración de agentes y alineamiento con marcos emergentes como la alianza E4S. Se propone una arquitectura de nodo modular con interfaces estandarizadas que permitan sustituir

componentes de distintos fabricantes sin alterar el sistema, cumpliendo la visión E4S.

4. Validación funcional en piloto: Desplegar un prototipo piloto de la aplicación VDC sobre un nodo edge y verificar su compatibilidad técnica con la infraestructura de medición existente basada en PRIME. Esto implica establecer y probar la comunicación bidireccional, hacia arriba con el sistema de telegestión (STG) y hacia abajo con los contadores inteligentes a través del nodo base, así como asegurar que el VDC ejecuta correctamente las tareas programadas de lectura y gestión como lo haría un concentrador físico. Adicionalmente, desarrollar un caso de uso avanzado que aproveche la computación en el borde, demostrando una funcionalidad de automatización (regulación de tensión OLTC) soportada por el VDC.

5. Estudio económico: Evaluar la viabilidad económica de escalar la solución de edge computing en la red de i-DE, comparando el escenario de nodos edge + VDC virtual frente al despliegue tradicional de concentradores físicos (TGUC), bajo el marco de remuneración regulada en España. Se analizan costes de inversión (CAPEX) y operación (OPEX) estimados para ambos escenarios, y se calculan indicadores financieros (VAN, TIR) considerando diferentes hipótesis regulatorias sobre la remuneración del software.

6. Estudio de escalabilidad: Evaluar la escalabilidad que ofrece el concentrador de datos virtual, sometiéndolo a una prueba de estrés para determinar el límite de puntos de suministros que es capaz de gestionar sin llegar a saturar los recursos del equipo.

Este proyecto es de gran relevancia en el contexto de las redes eléctricas inteligentes, ya que la validación de un concentrador virtual supone probar una pieza fundamental para la futura automatización distribuida en BT y la propuesta de una arquitectura modular para plataformas de edge computing permite la convivencia e incorporación de nuevos servicios avanzados o la rápida adaptación a nuevas normas regulatorias.

3. Análisis y estudio

3.1 Comparativa Barbara vs Onesait (Minsait)

En el proyecto se evaluaron dos plataformas industriales de edge computing probadas por i-DE, la solución de Barbara IoT y la de Minsait Onesait Phygital Edge. Ambas ofrecen un entorno para desplegar y gestionar aplicaciones en nodos edge ubicados en subestaciones, pero presentan diferencias arquitectónicas relevantes:

Minsait utiliza un Edge Management System (EMS) central instalado en Iberdrola i-DE que actúa como consola de gestión web y provee todos los servicios backend para administrar remotamente los edge nodes. Este EMS mantiene una conectividad bidireccional de plano de control con cada nodo a través de canales seguros, permitiendo a operadores realizar monitoreo del estado, despliegue de contenedores, actualizaciones de firmware/software, configuración, e incluso acceder por consola (SSH) a los nodos de forma remota. Adicionalmente, la arquitectura Onesait incorpora un gateway IoT multiprotocolo central (basado en un broker EMQX) que podría integrarse con sensores o RTUs, soportando protocolos field-to-center como MQTT-SN, CoAP, LwM2M, etc. Por otro lado, Barbara despliega un servidor de gestión de nodos conocido como Barbara Panel también dentro de la red corporativa de i-DE, el cual ofrece la interfaz de administración central, diseñado para operación en entornos aislados o sin conexión a Internet. El Panel de Barbara se instala en una máquina virtual en la DMZ industrial de i-DE, sirviendo como punto central de supervisión de los nodos edge. Ambos sistemas separan el plano de control del plano de datos, pero en Barbara todos los datos de campo se procesan en el nodo y solo se envían resultados agregados si es necesario, evitando actuar como gateway de todos los datos hacia el centro.

Tanto Onesait como Barbara utilizan mensajería MQTT segura para la comunicación entre gestor central y nodos, pero con distinta filosofía. En Onesait, el EMS emplea un broker MQTT (RabbitMQ) para intercambiar comandos y datos con los nodos de manera publish/subscribe, y además integra servicios avanzados como proxy inverso (Traefik), repositorio Git de configuraciones, registro privado de contenedores Docker, autenticación OAuth2, etc., todo para soportar múltiples casos de uso IoT desde la plataforma central. Barbara, por su parte, incluye en el Panel un broker MQTT(S) interno enfocado al plano de control, asumiendo que la gestión de datos de campo la realizan directamente las aplicaciones en cada nodo. En consecuencia, Barbara no incorpora en el Panel un sensor hub multiprotocolo, si no que delega en los nodos la conexión con contadores o sensores.

En resumen, Minsait/Onesait presenta una arquitectura de edge computing con un plano de control centralizado muy rico en servicios y posibilidades de integración central de datos IoT, ideal para escenarios corporativos amplios, ofreciendo un servicio completo y cerrado y tratando de alinearse con los requerimientos del E4S. Mientras que Barbara presenta una solución más abierta destinada a que sea el cliente quien personalice sus soluciones haciendo uso de su plataforma. Ambas plataformas demostraron cumplir requisitos clave (despliegue de contenedores, gestión remota, comunicaciones seguras MQTT/TLS), aunque cada opción tiene sus pros y contras. Las características de la plataforma de Minsait se alinean en mayor medida con las necesidades de Iberdrola, aunque no son es tan flexible a la hora de permitir la entrada a diferentes proveedores de aplicaciones como lo es Barbara, puesto que Minsait pretende ofrecer un servicio completo.

3.2 Despliegue On-Premise Plataforma Edge

Dada la estricta política de ciberseguridad de i-DE, se optó por desplegar la solución de Barbara IoT en modalidad on-premise dentro de la red corporativa OT de la compañía, en lugar de usar su versión cloud. Esto requirió adaptar el software Barbara originalmente ofrecido como SaaS a una implementación local cumpliendo los requisitos OT de Iberdrola. El equipo de Barbara, en coordinación con i-DE, realizó modificaciones personalizadas para instalar el Panel de Barbara en una máquina virtual del entorno de i-DE, asegurando el cumplimiento de los estándares IEC 62443 de seguridad industrial.

La arquitectura de despliegue se segmentó en tres zonas de red separadas por cortafuegos. La zona 1 o zona OT es donde residen los dispositivos de campo (nodos edge, nodo base PRIME, concentradores existentes, etc. en el CT o laboratorio), la segunda zona es una zona desmilitarizada (DMZ) industrial que aloja la máquina virtual con la plataforma Barbara (Panel y servicios asociados), y la tercera zona es la zona IT corporativa desde la cual los usuarios autorizados acceden vía sus PCs a la consola web del Panel. Esta segregación garantiza que el Panel de control es accesible solo internamente y que los nodos edge solo se comunican con el panel a través de la DMZ, evitando exposición directa a Internet.

Debido a que la plataforma Barbara originalmente asume conectividad a la nube para ciertas funciones (descarga de contenedores, actualizaciones, etc.), se llevaron a cabo varias acciones para operarla completamente offline, como precargar localmente en la VM todos los paquetes necesarios, imágenes Docker, actualizaciones de OS y certificados, transfiriéndolos de forma segura con supervisión del equipo de ciberseguridad de i-DE. Servicios clave como el registro de contenedores, el servicio OTA de actualizaciones y la autoridad certificadora se replicaron en la instancia local del Panel, de modo que el sistema funcionase sin depender de componentes externos. Asimismo, la comunicación con los nodos periféricos se aseguró en la capa de aplicación mediante TLS mutuo. Cada nodo posee un certificado de cliente específico del dispositivo y se conecta a los servicios de gestión del Panel a través de sesiones TLS cifradas y autenticadas mutuamente, utilizando nombres de dominio internos predefinidos. El servidor Panel se configuró con dos interfaces de red (una en la subred OT y otra en la DMZ) y con DNS específicos para enrutar adecuadamente el tráfico de administración, permitiendo que una sola instancia del Panel sirva a ambos ámbitos sin romper la segmentación.

Tras estas adaptaciones, el despliegue on-premise replicó con éxito las funcionalidades de la solución en la nube dentro del entorno segregado de i-DE. Se logró un sistema con comunicación segura de extremo a extremo y total control operativo local de los nodos edge y sus aplicaciones, sin comprometer la seguridad. Este entorno de prueba sentó las bases para instalar físicamente los nodos edge en el laboratorio de i-DE e integrarlos con los equipos reales del CT.

3.3 Arquitectura modular propuesta para E4S

E4S propone un modelo en capas para plataformas edge en subestaciones secundarias, buscando modularidad e interoperabilidad como atributos clave. Basándose en dicha referencia, se analizó la arquitectura E4S existente, mostrada en Ilustración 1 y se propusieron mejoras con el fin de ayudar a la definición de una arquitectura lo más interoperable posible.

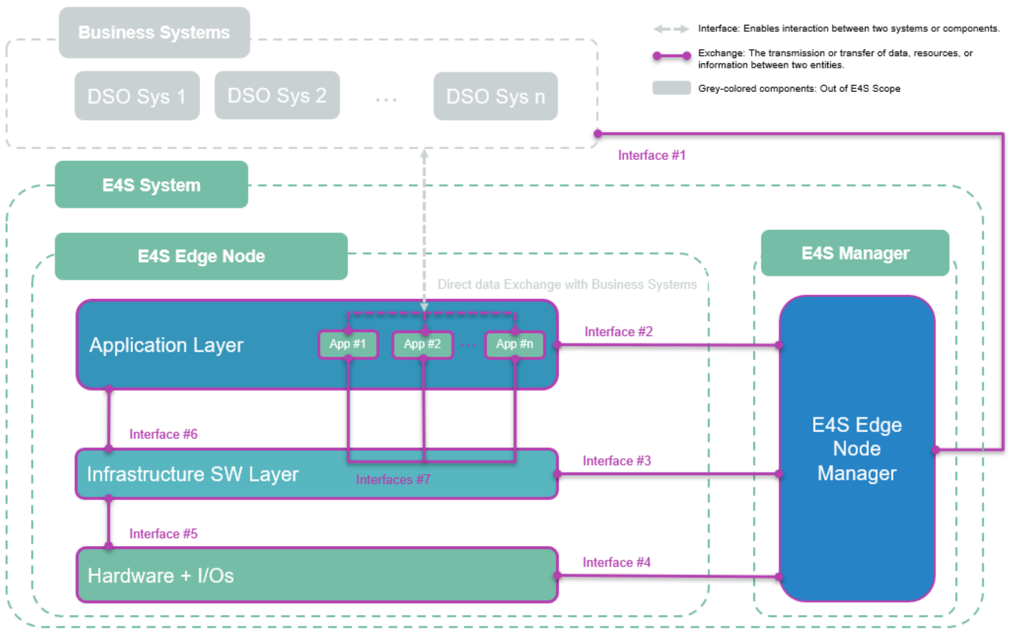


Ilustración 1: Esquema de la arquitectura propuesta por el E4S de [6].

En la concepción de E4S, el sistema se divide en dos subsistemas principales, por un lado el Node Manager (plataforma central de gestión en el edge), y por otro el Edge Node (dispositivo industrial desplegado en campo). El Edge Node a su vez consta de varias capas: la capa de hardware, que define el dispositivo físico y sus recursos de E/S; la capa de software de infraestructura (o capa de virtualización), que actúa como orquestador de los recursos hardware para las aplicaciones (incluye sistema operativo y software base que gestionan contenedores) y la capa de aplicación, donde residen las aplicaciones operativas del DSO normalmente encapsuladas en contenedores. El Node Manager, por su parte, supervisa todas estas capas en cada nodo y coordina su funcionamiento a nivel central.

La razón de adoptar una arquitectura en capas bien definidas es permitir despliegues masivos de nodos edge garantizando que cada capa cumple una función específica y que los componentes de cada nivel pueden interoperar y ser intercambiables sin dependencia de proveedor. Así, se busca evitar vendor lock-in y fomentar la estandarización y competitividad en los dispositivos, lo que a largo plazo reduce costes. Para lograr esto, E4S identifica un conjunto de 7 interfaces estandarizadas entre las capas y subsistemas, numeradas del #1 al #7. Cada interfaz especifica las comunicaciones o interacciones entre componentes adyacentes, promoviendo que dichos puntos de conexión sean abiertos y comunes para cualquier implementación.

El proyecto llevó a cabo primero un análisis detallado de las interfaces E4S definidas en la propuesta original, evaluando su propósito e importancia dentro de la operación del edge computing. Posteriormente, se plantearon ajustes y extensiones para mejorar la arquitectura de acuerdo con la experiencia obtenida en la implementación del VDC como separar claramente la capa de sistema operativo de la capa de agente de nodo (software de gestión local) dentro de la "capa de infraestructura" del nodo edge. E4S inicialmente agrupaba el OS y el agente en una misma capa, pero el diseño propuesto en este trabajo los divide en capas secuenciales distintas (OS abajo, Node Agent arriba), lo que mejora la claridad y permite definir mejor las responsabilidades y interfaces de cada uno. Esta separación introduce de hecho una nueva interfaz entre OS y agente de nodo (propuesta como interfaz #6 en nuestro esquema) que cubriría la interacción de bajo nivel. Además, se exploró la definición de la interfaz #7 e interfaz#8 (aún no especificadas por E4S).

La Ilustración 2 muestra el modelo planteado, incluyendo las interfaces E4S definidas y las nuevas o modificadas. En resumen, esta arquitectura modular facilita que un DSO pueda cambiar el "agente concentrador" (VDC) o añadir otros contenedores (p. ej. un agente de control de red BT) sin cambiar la plataforma subyacente, siempre que se respeten las interfaces estándar. Así se habilita que múltiples aplicaciones convivan en un mismo nodo edge y que distintos fabricantes puedan proveer componentes reemplazables (por ejemplo, un VDC de otro proveedor) cumpliendo los estándares de interoperabilidad.

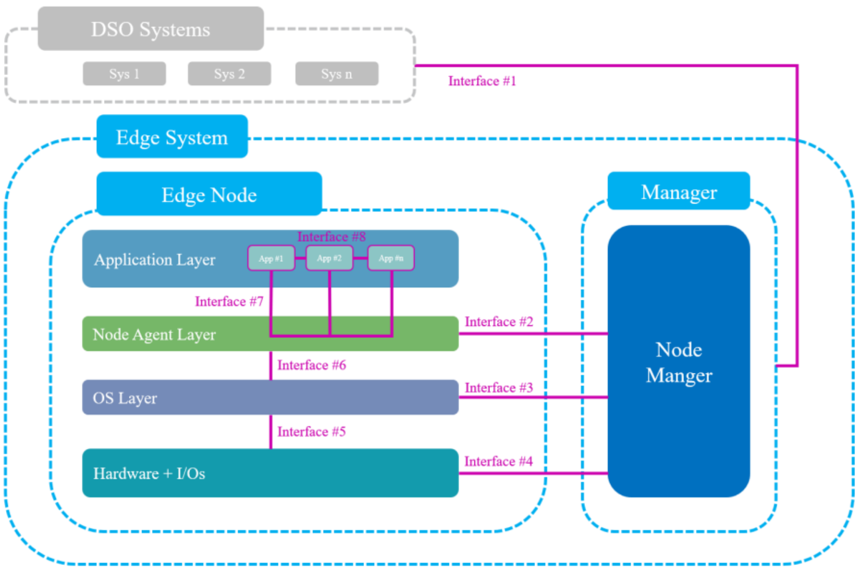
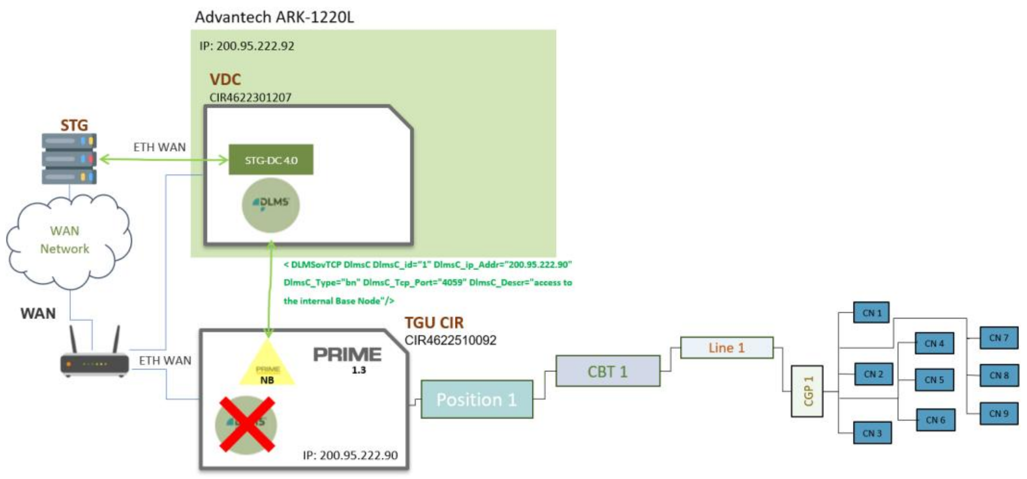


Ilustración 2: Esquema de la arquitectura propuesta en este proyecto.

3.4 Validación funcional, caso de uso OLTC y pruebas de escalabilidad

Una vez desplegada la infraestructura (plataforma Barbara on-premise y nodos edge con VDC), se procedió a validar exhaustivamente su funcionamiento mediante pruebas en laboratorio y pilotos en campo. Se consideraron tres ámbitos de validación: El primero consistió en pruebas funcionales de laboratorio del VDC, el segundo consistió en el desarrollo de un caso de uso avanzado OLTC implementado en un CT real, y el tercero en una prueba de escalabilidad conectando el VDC a múltiples CTs de campo simultáneamente. A continuación, se describen en detalle estos ensayos, junto con la arquitectura de conexión empleada en cada caso.

Pruebas funcionales en laboratorio: El objetivo inicial fue comprobar que el VDC virtual puede desempeñar todas las funciones básicas de un concentrador físico tradicional. Para ello, se conectó un nodo edge con el VDC a un entorno de laboratorio de i-DE compuesto por un nodo base PRIME (concentrador Circutor físico usado solo como módem PLC) y varios contadores inteligentes. La Ilustración 3 muestra esta configuración: el VDC (en el nodo edge) se comunicaba vía Ethernet con el nodo base, y a su vez con 9 contadores en baja tensión montados en el laboratorio, mientras que el Panel Barbara en la DMZ gestionaba el nodo.



```
< DLMSovTCP DlmsC DlmsC_id="1" DlmsC_ip_Addr="200.95.222.90"
DlmsC_Type="bn" DlmsC_Tcp_Port="4059" DlmsC_Descr="access to
the internal Base Node"/>
```

Ilustración 3: Esquema de conexión del VDC en el laboratorio de Iberdrola i-DE.

Se definió una batería de tres pruebas secuenciales:

1. Test de conectividad y configuración: consistente en solicitar desde el sistema central un informe S12 al VDC para verificar su configuración de fábrica y después enviar una orden B07 de parametrización inicial (estableciendo las IPs del STG y FTP igual que se haría con un concentrador nuevo). Previamente se envió una B07 al concentrador físico para deshabilitar su propio servicio interno y usarlo como simple nodo base.

2. Test de lecturas de medida: consistente en comprobar que el VDC puede leer datos de contadores periódicamente. Se simularon órdenes de lectura (por ejemplo S24/G24 de descubrimiento de equipos en la red PRIME) y consultas de medidas (reportes S21 de perfiles de carga) para confirmar que el VDC recibe los datos de los contadores a través del nodo base y los reenvía al sistema de telegestión (STG) o los almacena en FTP, según corresponda.

3. Test de tareas programadas: consistente en verificar que el VDC ejecuta de forma autónoma tareas periódicas (como haría un concentrador físico al recoger lecturas diarias, cierres de facturación, etc.). Se programaron tareas típicas y se comprobó mediante registros de eventos que el VDC las lanzaba en tiempo y forma, interactuando con los contadores adecuadamente.

Caso de uso OLTC en campo: Para demostrar el potencial de tener aplicaciones avanzadas corriendo en el edge, se desarrolló un caso de uso de regulación de tensión en carga (On-Load Tap Changer) en un transformador inteligente. La prueba se realizó en un CT real en el norte de España equipado con un transformador con OLTC. En dicho CT, el concentrador físico existente (un ZIV) se utilizó como nodo base PLC, se deshabilitó su funcionalidad de concentrador y se conectó remotamente el VDC del laboratorio al nodo base de ese CT. La Ilustración 4 muestra esta arquitectura, donde el nodo edge con VDC permaneció en el laboratorio de i-DE (Madrid), comunicándose por la red de i-DE con el CT remoto. Este primer escenario tuvo como fin implementar localmente en el edge un algoritmo que tomara decisiones de cambio de toma del transformador en base a mediciones de tensión de los contadores, sin depender de la nube ni de intervención humana, logrando control en tiempo casi real.

El CT utilizado para el desarrollo del caso de uso contaba con 175 contadores, de los cuales se realizó una preselección de 5 contadores representativos para monitorear. Los criterios de selección incluyeron: escoger solo contadores trifásicos (400 V) ya que están asociados a cargas mayores o actúan como supervisores de varios monofásicos; tomar al menos un contador de cada una de las 5 salidas de BT del transformador para cubrir toda el área; excluir contadores con comunicación poco fiable; y priorizar aquellos que alimentaran a muchos usuarios. Teniendo en cuenta estos criterios de selección se identificaron los 5 equipos óptimos y se desarrolló una aplicación Python contenerizada en el nodo edge que implementara la lógica de control OLTC. Esta aplicación se diseño para que periódicamente solicitara al VDC lecturas S21 de perfiles de carga de los cinco contadores (simulando una orden periódica del sistema central), obtuviese las tensiones de fase medidas, y calculase el promedio de tensión trifásica de los puntos supervisados. Si el promedio se desvía del valor nominal (230 V) más allá de un banda de tolerancia de ±3%, el algoritmo decide un ajuste de toma decidiendo si se debe aumentar, mantener o bajar la toma. Esta decisión se registra como un comando de tap up/down/no-change. El algoritmo corre cada 5 segundos, y cada ciclo almacena en una base de datos de series temporales InfluxDB (instalada en el nodo edge) tanto los valores de tensión de cada contador como la orden de tap calculada. Paralelamente se desplegó en el edge un contenedor con Grafana que consulta la base Influx y visualiza en tiempo real las tensiones y las decisiones de tap. La Ilustración 4 muestra esquemáticamente cómo el VDC virtual lee datos de los contadores (vía nodo base), el script de control procesa dichas lecturas y decide la acción de OLTC, los datos se guardan en InfluxDB, y Grafana presenta las tendencias de voltaje y comandos de tap, todo ello dentro del nodo edge. Esta arquitectura distribuida permite realizar monitorización y control en tiempo real directamente en el CT, sin depender de la latencia de comunicar cada lectura al centro.

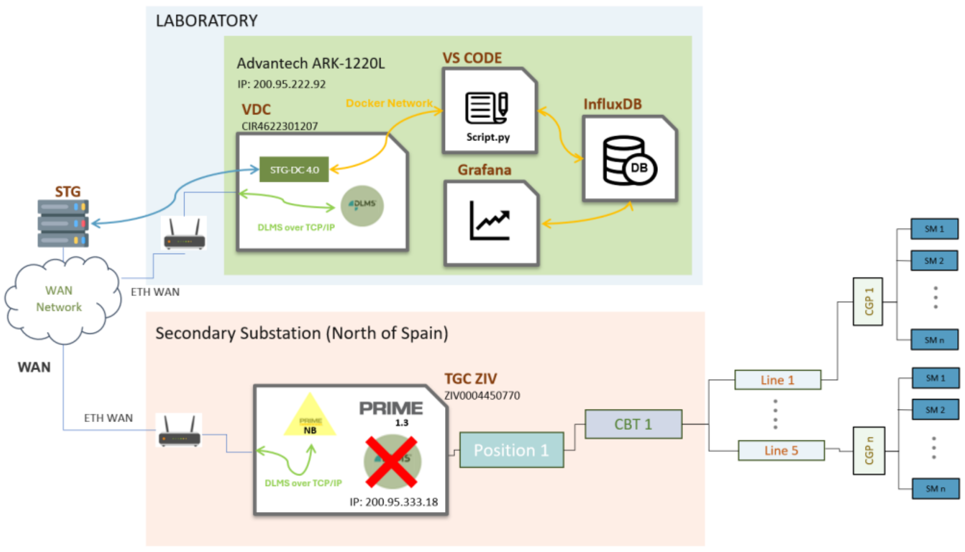


Ilustración 4: Esquema de las conexiones físicas y lógicas entre el VDC y el CT de campo y de las aplicaciones edge.

Prueba de escalabilidad: En este escenario se evaluó la capacidad del VDC para gestionar múltiples subestaciones en paralelo, empujándolo a su límite teórico. En la prueba se conectaron 12 centros de transformación reales de la zona urbana de Madrid, seleccionadas entre las de mayor número de abonados (~900 contadores cada una). Todos estos CTs se vincularon simultáneamente a la instancia de VDC corriendo en un nodo edge del laboratorio. La Ilustración 5 representa esta configuración, donde el VDC actúa como concentrador virtual multi-CT.

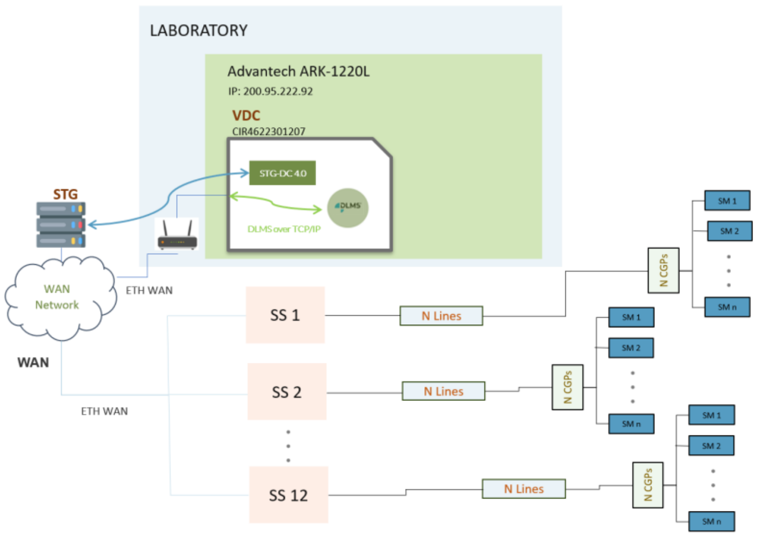


Ilustración 5: Esquema de conexión para la prueba de escalabilidad del VDC.

4. Resultados

4.1 Validación funcional del VDC

Las pruebas de validación funcional confirmaron que el VDC desplegado en el nodo edge reproduce correctamente todas las funciones esenciales de un concentrador físico, con pequeñas salvedades solucionables. En la Tabla 1 se recogen los principales ensayos realizados y su resultado.

Categoría del Test	*Test*	*Satisfactorios*	*Recibidos en STG o FTP*
Configuración y Conectividad	S12 – Factory Configuration Report	✓	✗ *(Missing fields: ipCom2, ipMask2)*
	B07 – Disable Internal DLMS	✓	–
	B07 – Initial Configuration	✓	–
Medidas	S24/G24 – Meter Discovery	✓	✓
	S21 – Meter 1 (No Load)	✓	✓
	S21 – Meter 7 (Load: 60 W)	✓	✓
Validación de Tareas Programadas	S05 – Daily Billing Report	✓	✗ *(STG development issue)*
	S02 – Hourly Load Profile	✓	✓
	S04 – Monthly Billing Closure	✓	✓ *(STG development issue)*
	S09 – Meter Event Log	✓	✓
	T01 – Reboot Order	✓	✓
	T07 – Forced Time Sync	✓	✓
	S17 – Concentrator Event Log	✓	✓
	S24/S11 – SM List and BN Status	✓	✗ (S11 only: No BN)
	S14 – Voltage and Current Profile	✓	✗ (no LVS)

	G01 – Communication Statistics (Hourly)	✓	✓
	G02 – Communication Statistics (Daily)	✓	✓
	G12 – DC Performance History	✓	✓

Tabla 1: Resumen de resultados de las pruebas de validación del VDC.

En resumen, el VDC pudo recibir y responder a comandos de configuración (S12) y órdenes de parametrización (B07), descubrir contadores en la red PRIME (S24/G24), consultar medidas horarias (S21) e ingresar datos en el sistema central, así como ejecutar tareas programadas diarias, horarias y eventuales de forma autónoma. Todos los casos de prueba obtuvieron resultado satisfactorio en cuanto a ejecución en el VDC, y en la mayoría de ellos el reporte llegó al sistema de telegestión o almacenamiento destino como se esperaba. Por ejemplo, se verificó que el VDC tras una orden S24 detectó correctamente todos los contadores conectados al nodo base, y que ante consultas S21 de lectura, los valores se almacenaron en el servidor FTP. Asimismo, en las tareas periódicas se observó que el VDC iniciaba puntualmente cada petición y transmitía las respuestas de los contadores.

Algunos incidentes menores se identificaron durante estas pruebas, pero no atribuibles a fallos del VDC sino a limitaciones del entorno de pruebas. Por ejemplo, el informe S12 inicial fue rechazado por el RMS debido a campos obligatorios ausentes (ipCom2, ipMask2) en la configuración, lo cual se solucionaría fácilmente ajustando la especificación para que esos campos no sean requeridos o rellenándolos con valores por defecto. También, en las tareas programadas, ciertos reportes como S05 y S17 no se almacenaron en el STG de desarrollo por problemas conocidos de ese entorno, que no ocurrirían en el entorno de producción actualizad0. En todo caso, no se hallaron impedimentos técnicos de fondo, el VDC demostró ser viable técnicamente para sustituir concentradores físicos, requiriendo apenas adaptaciones menores de la especificación existente. Este es un resultado crucial, pues despeja las dudas sobre la funcionalidad básica del VDC en la infraestructura de i-DE. En conclusión, las pruebas de laboratorio validaron la viabilidad de desplegar VDCs en sustitución de DCs físicos, cumpliendo con las funcionalidades esperadas una vez solucionados detalles de configuración.

4.2 Resultados caso de uso OLTC

En el piloto OLTC se evaluó la capacidad del VDC para habilitar un control en tiempo real en el edge. Durante la prueba en campo (CT con i-Trafo), el algoritmo de control en el nodo edge estuvo activo durante varias horas, monitoreando continuamente las tensiones de los 5 contadores seleccionados y tomando decisiones sobre la toma del transformador. A través de la interfaz Grafana se observaron en tiempo real tanto las tensiones trifásicas medidas en cada punto, como la decisión de ajuste de toma calculada en cada iteración. La Ilustración 6 como inicialmente el transformador estaba en posición nominal de toma, con tensiones dentro de rango. Luego, ante un aumento de tensión en ciertas fases más allá del 3 % de tolerancia, el algoritmo detectó la desviación y comenzó a emitir comandos para bajar el toma con el fin de reducir la tensión suministrada. En las gráficas se ve cómo tras superarse el umbral superior, el algoritmo repetidamente recomienda bajar un paso el toma.

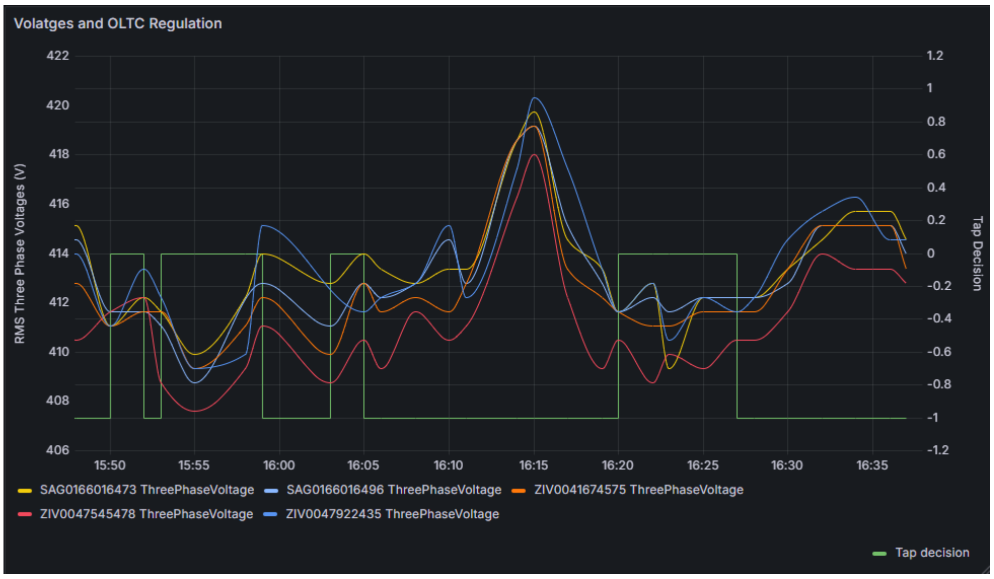


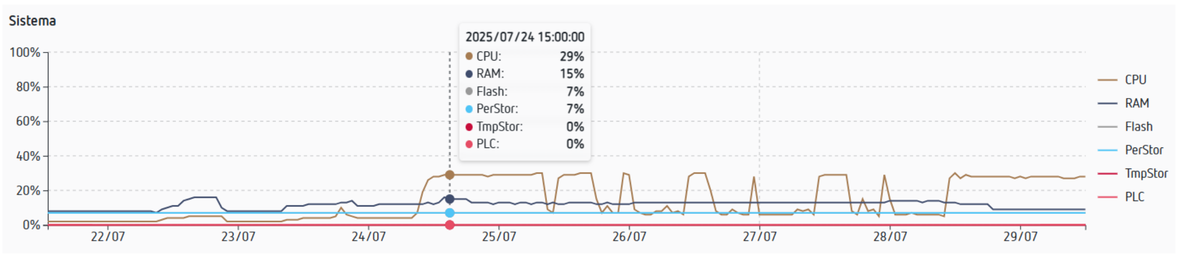
Ilustración 6: Decisión de la toma del i-Trafo y los perfiles de tensión trifásica de los contadores inteligentes.

Es importante señalar que, por motivos operativos y de seguridad, no se aplicaron físicamente los comandos al OLTC real del transformador durante el piloto. Es decir, el sistema no ejecutó el cambio de toma en el aparato, sino que solo lo simuló registrándolo. Debido a esta falta de cierre del lazo de control, las tensiones medidas permanecieron elevadas y el algoritmo continuó emitiendo la misma recomendación en cada ciclo. A pesar de ello, el piloto demuestra que el VDC y la aplicación edge son capaces de obtener medidas en tiempo real de los contadores, procesarlas localmente con un algoritmo personalizado, y generar acciones de control con muy baja latencia. Todo el proceso ocurrió en el nodo edge, confirmando la viabilidad de implementar control distribuido en el CT usando la infraestructura de VDC. Los resultados visualizados en Grafana mostraron cómo el sistema reaccionaría dinámicamente a las condiciones de tensión. Este caso de uso sienta un precedente para aplicaciones avanzadas en el edge, que podrán añadirse sobre la plataforma modular.

4.3 Resultados de escalabilidad

La prueba de escalabilidad confirmó que la solución propuesta puede manejar un volumen de dispositivos muy superior al de los despliegues actuales de concentradores individuales. El VDC virtual logró mantener comunicación simultánea con ~6000 contadores distribuidos en 12 CTs distintos, lo cual es equivalente a concentrar en un solo nodo edge el trabajo de 12 concentradores físicos. Este resultado sugiere que en un futuro podría ser posible centralizar por zonas o agrupaciones varios CTs en un mismo nodo edge potente, reduciendo la cantidad de equipos necesarios en campo.

Un hallazgo fue que el límite práctico durante la prueba estuvo dado por temas de seguridad de los contadores, no por saturación del VDC. Las estadísticas de uso de recursos mostradas en la Ilustración 7 permiten ver que el nodo edge aún tenía margen en CPU, memoria y red al manejar ~5900 conexiones. Además, aunque solo se pudieron obtener datos de los contadores no asegurados, el VDC sí envió peticiones a todos los contadores. Es decir, el software escaló hasta ese número de sesiones sin bloquearse ni colapsar la red. En condiciones reales (con un STG capaz de gestionar las claves), se esperaría que el VDC atendiera a todos los contadores con éxito. De hecho, se comprobó que algunas tareas broadcast (como una orden S05 general) fueron enviadas pero fallaron solo por el tema de active key pending en muchos de ellos. Esto evidencia la robustez del VDC para programar y manejar múltiples hilos de peticiones concurrentes.



2025/07/24 15:00:00
CPU: 29%
RAM: 15%
Flash: 7%
PerStor: 7%
TmpStor: 0%
PLC: 0%

Ilustración 7: Consumo másximo de recursos de hardware por parte del VDC durante la prueba de escalabilidad.

4.4 Impacto Económico

El análisis económico realizado tiene como objetivo comparar dos escenarios de despliegue para la modernización de los centros de transformación (CT) de Iberdrola i-DE: el escenario tradicional con despliegue de nuevos concentradores físicos TGUC frente al escenario innovador con nodos edge que ejecutan un concentrador virtual (VDC). Para esta comparación se han utilizado las hipótesis de inversión inicial (CAPEX) y costes operativos (OPEX) recogidas en las tablas correspondientes Tabla 2 y Tabla 3.

Equipo	***Coste***	***Unidad***
TGUC	450.00	€/Un
LVS	70.00	€/Un
Nodo edge	300.00	€/Un
NB PRIME	100.00	€/Un
Licencia VDC	250.00	€/Instancia
Licencia DPP	175.00	€/nodo
Instalación en el CT	170.00	€/CT
Despliegue Plataforma Edge	445,000.00	€

Tabla 2: CAPEX

Concepto	***Valor***	***Unidad***
Coste Intervención en CTs	140.00	€/CT

Intervenciones de O&M	3%	% de CTs al año
Reducción en intervenciones Edge	30%	% de intervenciones al año

Tabla 3: OPEX

Considerando el número total de CT afectados, especificado en la Tabla 4, se ha asumido un despliegue gradual y realista descrito mediante una curva tipo Weibull Ilustración 8. A partir de estas hipótesis se ha realizado un análisis coste-beneficio detallado para cada escenario durante el horizonte temporal del proyecto (15 años), cuyos resultados pueden observarse visualmente en las gráficas correspondientes Ilustración 9 y Ilustración 10.

Clasificación	***CTs***	***CTs con 2 o más posiciones***	***Posiciones MT/BT***
Total	102,054.00	22,314.00	124,368.00
Urbanos	55,819.00	22,314.00	78,133.00
Rurales	26,185.00	0.00	26,185.00
Areas rurales distribuidas	20,050.00	0.00	20,050.00

Tabla 4: Número de CTs de MT/BT en Iberdrola i-DE.

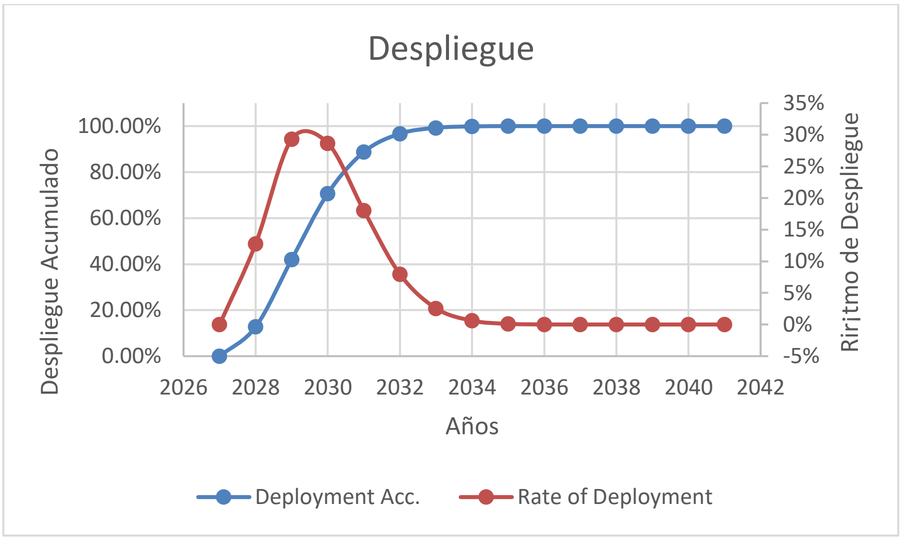


Ilustración 8: Curva de despliegue de TGUCs y nodos edge siguiendo una distribución de Weibull.

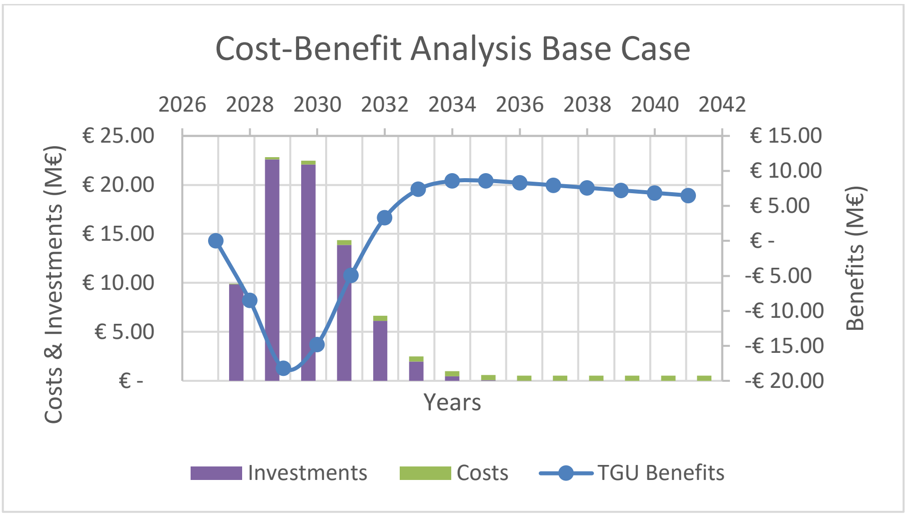


Ilustración 9: Análisis de coste-beneficio del caso base.

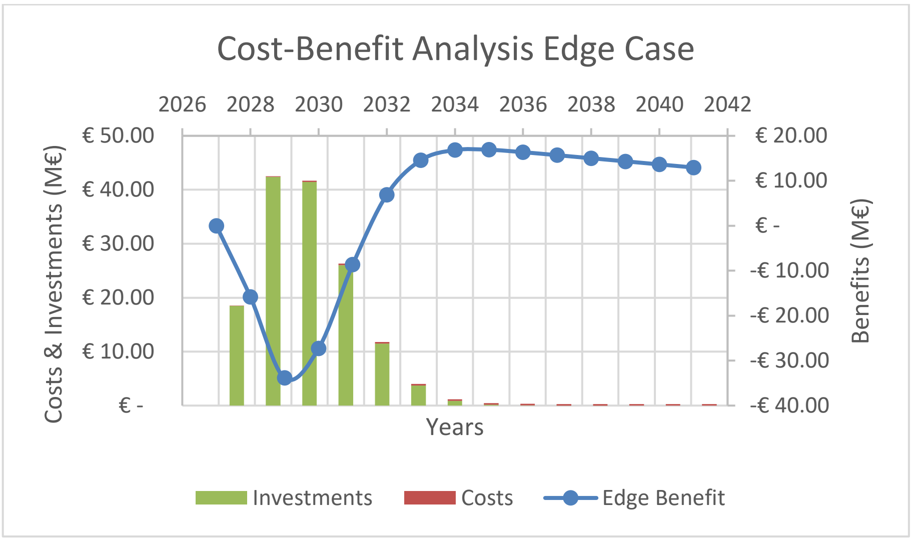


Ilustración 10: Análisis de coste-beneficio del caso edge (100% de remuneración)

Los indicadores económicos clave calculados para cada escenario incluyen el Valor Actual Neto (VAN) y la Tasa Interna de Retorno (TIR). Estos indicadores han sido evaluados bajo tres supuestos distintos sobre la remuneración regulatoria del coste del software: remuneración completa (100%), remuneración nula (0%) y la remuneración mínima necesaria para hacer más ventajoso el despliegue de nodos edge frente al caso base TGUC. Los resultados resumidos de estos cálculos se presentan en la Tabla 5.

ANÁLISIS C&B	*VAN*	*TIR*
Caso Base	5,234.11 €	6.462 %
Caso Edge (50 % Remuneración)	-12,947,412.67 €	4 %
Caso Edge (0 % Remuneración)	-31,080,257.63 €	-1 %
Caso Edge (86 % Remuneración)	5,234.11 €	6.462 %

Tabla 5: Resumen del NPV y el TIR en función del porcentaje de remuneración.

Como conclusión principal del análisis económico, cabe destacar que dado el actual marco regulatorio en España, que favorece la inversión en activos físicos (CAPEX), se ha determinado que la opción de nodos edge con VDC resulta económicamente más ventajosa que el escenario base de TGUC únicamente si el regulador remunera el coste del software al 86% o superior. Este umbral representa el punto crítico a partir del cual la alternativa edge computing es preferible desde una perspectiva de rentabilidad financiera.

5. Conclusiones

Este trabajo ha demostrado la viabilidad técnica de virtualizar el concentrador de datos (VDC) en entornos distribuidos, validando su funcionamiento en escenarios reales de campo. Los resultados de las pruebas funcionales, de escalabilidad y del caso de uso de regulación OLTC confirman que la solución es capaz de replicar las funciones clave del concentrador físico, con flexibilidad adicional para desplegar aplicaciones inteligentes en el edge. La arquitectura modular propuesta aporta un enfoque escalable e interoperable alineado con las iniciativas de estandarización como E4S, facilitando la integración de agentes distribuidos de distintos proveedores. Desde un punto de vista económico, el análisis coste-beneficio ha mostrado que, bajo determinadas condiciones regulatorias, el despliegue de nodos edge con VDC puede ser una alternativa rentable frente a los concentradores físicos tradicionales.

En conjunto, el proyecto sienta una base sólida para avanzar hacia redes más inteligentes, virtualizadas y adaptadas a la transición digital del sistema eléctrico.

VIRTUAL DATA CONCENTRATOR: TESTING, VALIDATION, AND DOCUMENTATION OF CONTAINERIZATION/VIRTUALIZATION ARCHITECTURE FOR TELE MANAGEMENT DATA CONCENTRATOR APPLICATION IN DISTRIBUTED ENVIRONMENTS

Author: Valdivielso Suárez, Emilio Juan.
Director: Sánchez Paniagua, Aurelio.
Collaborating Entity: Iberdrola i-DE

ABSTRACT

This project aims to technically and functionally validate the virtualization of the Data Concentrator (VDC) in distributed power grid environments, as an alternative to traditional embedded hardware. A comparative analysis was conducted between Minsait's previously tested platform and Barbara's edge computing solution, currently under evaluation at Iberdrola i-DE, as well as against the conventional embedded concentrator. The study assessed their ability to support critical substation applications. A complete deployment of a Barbara-based edge node was carried out on industrial-grade hardware. Functional equivalence with the embedded solution was demonstrated through tests involving configuration, interconnectivity, data acquisition, and scheduled task execution. Additionally, a voltage regulation use case using On-Load Tap Changers (OLTC) was developed and implemented directly on the virtualized node, showcasing the platform's ability to support new grid automation functionalities. Scalability tests confirmed the VDC's capacity to manage over 7,000 smart meters without compromising operational integrity. An economic analysis further highlighted the feasibility of the virtualized approach under specific regulatory frameworks. Beyond technical validation, this work contributes to the E4S Alliance by proposing interface definitions that enhance interoperability among edge computing components. The results lay the groundwork for modular, scalable, and interoperable architectures in future smart grid deployments.

Keywords: Distributed Processing Platforms (DPP), Edge Computing, Edge Node, Intelligent Secondary Substations, Interoperability, Smart Grids, Virtualized Data Concentrator.

1. Introduction

The electrical distribution network is undergoing a profound digitalization process, driven by the need to improve efficiency, integrate distributed renewable energies (DERs), and meet decarbonization goals. According to Iberdrola "Smart Grids are electricity networks that can intelligently and dynamically integrate the actions of all the users connected to them – those that generate energy, those that consume energy or those that do both – in order to supply electricity efficiently, sustainably, economically and safely" [1]. The evolution toward this model in distribution networks seeks to optimize reliability and efficiency through real-time monitoring, bidirectional energy flows, and adaptive control. In this context, the modernization of Secondary Substations (SS) plays a crucial role, as they are key nodes that connect the medium-voltage network with the low-voltage network and can provide new functionalities such as DER integration, predictive maintenance, dynamic load management, and the processing of measurement data without the need to send it to the cloud.

Currently, many of these functions in Secondary Substations (SS) depend on proprietary hardware devices such as Data Concentrators (DC) for remote management and Remote Terminal Units (RTUs) dedicated to supervision and remote control. This device-specific approach entails significant limitations, such as the difficulty in finding alternative suppliers (due to highly customized hardware and software), very rigid development and deployment cycles (incorporating new functionalities can take several years), and a lack of flexibility to adapt to regulatory or technological changes. Although the current solution implemented by the distributor Iberdrola i-DE has achieved large-scale deployment, managing more than 120,000 transformation positions and approximately 11.6 million meters, it does not offer sufficient adaptability to meet new innovation or regulatory demands. This is due to the monolithic nature of existing systems and the need to physically update thousands of devices to implement any improvement.

In response, the current trend points toward the use of unified IED-type (Intelligent Electronic Device) platforms on generic hardware within the Secondary Substation, integrating the functions of concentrator, RTU, and low-voltage supervisor through virtualization or containerization. Although standards such as IEC 61850 have improved interoperability in smart substations, they do not fully resolve the aforementioned limitations. [2]. For this reason, the combination of virtualization and edge computing emerges as the main solution to this problem. Edge computing involves decentralizing data processing to local nodes, reducing latency and enabling real-time responses for applications such as load balancing, fault detection, or immediate data analysis, thereby facilitating the operation of the low-voltage network. On the other hand, virtualization decouples software from hardware, making it possible for traditional embedded solutions to run on generic devices. This allows for the unification of functions previously distributed across multiple devices into a single node, reducing capital costs and improving operational flexibility. A key initial use case is the virtual data concentrator (VDC), given its importance in collecting and providing data used by other applications. By virtualizing the concentrator, it can more easily adapt to the evolution of protocols and incorporate new functionalities or regulatory requirements without physically replacing thousands of devices.

To enable this vision, various edge application orchestration platforms have emerged. Industrial solutions such as Barbara IoT [3] or Onesait Phygital Edge by Minsait, among other distributed processing platforms mentioned in [4], allow for the secure and scalable deployment of software on heterogeneous edge nodes. These platforms provide mechanisms for centralized management of nodes and containers, secure communications, and remote application deployment. In line with these advances, the Spanish technological platform Futured (which brings together Iberdrola and other electric utilities) proposes transforming the traditional Secondary Substation into an Intelligent and Automated Secondary Substation with Computing and Telecommunications Infrastructure (SASCTI) [5]. n parallel, the E4S (Edge for Smart Secondary Substations) alliance aims to define common standards and architectures that facilitate collaboration between Distribution System Operators (DSOs) and manufacturers, promoting modular and vendor-agnostic solutions instead of monolithic systems [6].

In this context of modernization, the distributor Iberdrola i-DE has launched the PRADA project, which during the next regulatory period (2026–2031) aims to update its metering infrastructure, including the migration from PRIME 1.3.6 to PRIME 1.4 and the renewal of old meters to version SM 2.0. These changes offer the opportunity to also modernize the data concentrators in the secondary substations, replacing obsolete equipment (such as concentrators with integrated 3G modems) with more advanced solutions. Specifically, the project proposes comparing two scenarios. The first consists of the traditional deployment of new physical concentrators (referred to as TGUC) with dedicated hardware (including an integrated PRIME base node and LV supervisor) at each transformer position. The second scenario involves installing edge nodes in the secondary substations running virtualized applications, with the VDC being the first of them. Under this second approach, a single industrial node per substation, connected to external modules (PRIME base node and LV supervisor), could replace multiple proprietary concentrators. Iberdrola i-DE aims to investigate and validate the technical feasibility of this concentrator virtualization, expecting thereby to improve the flexibility, interoperability, and scalability of the network, as well as to assess its economic viability within the current Spanish regulatory framework.

2. Project definition

The main objective of this project is to technically validate a Virtual Data Concentrator (VDC) implemented on an edge platform, and to propose a modular architecture that facilitates its integration into distributed and multi-vendor environments aligned with the E4S initiative. To achieve this general objective, several specific sub-objectives were defined:

1. Edge platform analysis: Critically study the Barbara IoT platform in a laboratory setting, evaluating its architecture, its capacity to orchestrate applications on distributed nodes, and its performance under real operating conditions. A technical comparison was also carried out with another edge computing platform (Onesait by Minsait) to identify strengths, limitations, and suitability for smart grid requirements.

2. Deployment documentation: Document the complete process of implementing the VDC in an OT environment, including the installation of the Barbara OS operating system on generic industrial hardware and the configuration of the edge node. This includes the adaptation of the Barbara platform to i-DE's infrastructure and the connection with field devices.

3. Modular and interoperable architecture: Define the requirements and architecture that a Distributed Processing Platform (DPP) must meet to be vendor-agnostic, with an emphasis on interoperability, agent migration, and alignment with emerging frameworks such as the E4S alliance. A modular node architecture is proposed, with standardized interfaces that allow components from different manufacturers to be replaced without altering the system, in line with the E4S vision.

4. Functional validation in pilot: Deploy a pilot prototype of the VDC application on an edge node and verify its technical compatibility with the existing measurement infrastructure based on PRIME. This involves establishing and testing bidirectional communication, northbound with the remote management system (RMS) and southbound with the smart meters through the base node, as well as ensuring that the VDC correctly executes the scheduled reading and management tasks as a physical concentrator would. Additionally, develop an advanced use case that leverages edge computing, demonstrating an automation functionality (OLTC voltage regulation) supported by the VDC.

5. Economic study: Evaluate the economic feasibility of scaling the edge computing solution within the i-DE network, comparing the scenario of edge nodes + virtual VDC against the traditional deployment of physical concentrators (TGUC), under the regulated remuneration framework in Spain. Estimated capital expenditures (CAPEX) and operational expenditures (OPEX) for both scenarios are analyzed, and financial indicators (NPV, IRR) are calculated considering different regulatory assumptions regarding software remuneration.

6. Scalability study: Evaluate the scalability offered by the data concentrator by subjecting it to a stress test to determine the limit of supply points it can manage without saturating the device's resources.

This project is highly relevant in the context of smart grids, as the validation of a virtual concentrator represents the testing of a fundamental component for future distributed automation in low-voltage networks, and the proposal of a modular architecture for edge computing platforms enables the coexistence and incorporation of new advanced services or the rapid adaptation to new regulatory standards.

3. Analysis and study

3.1 Barbara vs Onesait (Minsait) Comparison

As part of the project, two industrial edge computing platforms tested by i-DE were evaluated: the solution from Barbara IoT and Minsait's Onesait Phygital Edge. Both offer an environment for deploying and managing applications on edge nodes located in substations, but they present relevant architectural differences:

Minsait uses a central Edge Management System (EMS) installed at Iberdrola i-DE, which acts as a web management console and provides all backend services for remotely managing the edge nodes. This EMS maintains bidirectional control-plane connectivity with each node through secure channels, allowing operators to monitor status, deploy containers, perform firmware/software updates, configure settings, and even access the nodes remotely via console (SSH). Additionally, the Onesait architecture incorporates a central multiprotocol IoT gateway (based on an EMQX broker) that could be integrated with sensors or RTUs, supporting field-to-center protocols such as MQTT-SN, CoAP, LwM2M, etc.

On the other hand, Barbara deploys a node management server known as the Barbara Panel, also within i-DE's corporate network, which provides the central administration interface and is designed for operation in isolated or offline environments. The Barbara Panel is installed on a virtual machine in i-DE's industrial DMZ, serving as the central monitoring point for edge nodes. Both systems separate the control plane from the data plane, but in Barbara all field data is processed on the node, and only aggregated results are sent if necessary, avoiding acting as a gateway for all data to the center.

Both Onesait and Barbara use secure MQTT messaging for communication between the central manager and the nodes, but with different philosophies. In Onesait, the EMS uses an MQTT broker (RabbitMQ) to exchange commands and data with the nodes in a publish/subscribe manner, and also integrates advanced services such as a reverse proxy (Traefik), a Git repository for configurations, a private Docker container registry, OAuth2 authentication, etc., all to support multiple IoT use cases from the central platform.

Barbara, for its part, includes an internal MQTT(S) broker in the Panel focused on the control plane, assuming that field data management is handled directly by the applications on each node. Consequently, Barbara does not include a multiprotocol sensor hub in the Panel, but instead delegates the connection to meters or sensors to the nodes themselves.

In summary, Minsait/Onesait presents an edge computing architecture with a centralized control plane rich in services and possibilities for central IoT data integration, ideal for large-scale corporate scenarios, offering a complete and closed service while aiming to align with E4S requirements. Barbara, on the other hand, presents a more open solution intended for the client to customize their solutions using the platform. Both platforms demonstrated compliance with key requirements (container deployment, remote management, secure MQTT/TLS communications), although each option has its pros and cons. The features of the Minsait platform are more aligned with Iberdrola's needs, although it is not as flexible in allowing the entry of different application providers as Barbara is, since Minsait aims to offer a complete service.

3.2 On-Premise Deployment of the Barbara platform

Given i-DE's strict cybersecurity policy, the Barbara IoT solution was deployed in an on-premise mode within the company's OT corporate network, instead of using its cloud version. This required adapting the Barbara software, originally offered as SaaS, to a local implementation that meets Iberdrola's OT requirements. The Barbara team, in coordination with i-DE, carried out custom modifications to install the Barbara Panel on a virtual machine within i-DE's environment, ensuring compliance with the IEC 62443 industrial security standards.

The deployment architecture was segmented into three network zones separated by firewalls. Zone 1, or the OT zone, is where the field devices reside (edge nodes, PRIME base node, existing concentrators, etc., in the substation or laboratory); the second zone is an industrial demilitarized zone (DMZ) that hosts the virtual machine with the Barbara platform (Panel and associated services); and the third zone is the corporate IT zone from which authorized users access the Panel's web console via their PCs. This segregation ensures that the control Panel is accessible only internally and that the edge nodes communicate with the Panel exclusively through the DMZ, avoiding direct exposure to the Internet.

Since the Barbara platform originally assumes cloud connectivity for certain functions (container downloads, updates, etc.), several actions were carried out to operate it fully offline, such as preloading all necessary packages, Docker images, OS updates, and certificates locally into the VM, transferring them securely under the supervision of i-DE's cybersecurity team. Key services such as the container registry, the OTA update service, and the certificate authority were replicated in the local instance of the Panel, so that the system could operate without relying on external components. Likewise, communication with the edge nodes was secured at the application layer using mutual TLS. Each node holds a device-specific client certificate and connects to the Panel's management services over encrypted and mutually authenticated TLS sessions, using predefined internal domain names. The Panel server was configured with two network interfaces (one in the OT subnet and another in the DMZ) and with specific DNS and routing settings to properly steer management traffic, allowing a single instance of the Panel to serve both domains while preserving network segmentation.

After these adaptations, the on-premise deployment successfully replicated the functionalities of the cloud solution within i-DE's segregated environment. A system was achieved with secure end-to-end communication and full local operational control of the edge nodes and their applications, without compromising security. This test environment laid the groundwork for the physical installation of the edge nodes in i-DE's laboratory and their integration with the actual equipment of the Secondary Substation.

3.3 Proposed Modular Architecture for E4S

E4S proposes a layered model for edge platforms in secondary substations, aiming for modularity and interoperability as key attributes. Based on this reference, the existing E4S architecture, shown in Illustration 1, was analysed, and improvements were proposed in order to contribute to the definition of an architecture as interoperable as possible.

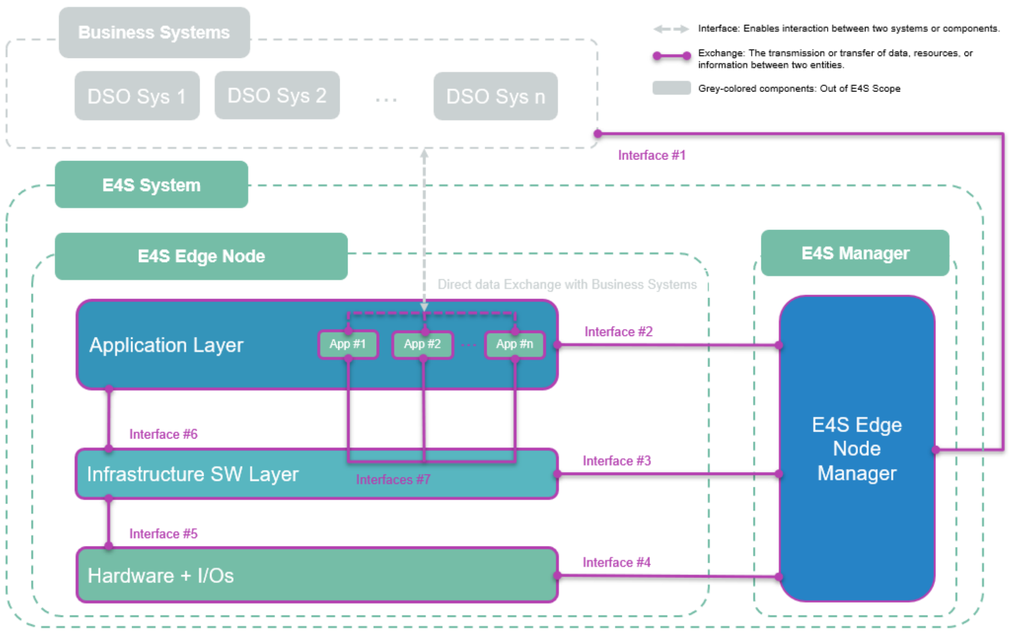


Illustration 1: Schematic of the proposed architecture by the E4S from [6].

In the conception of E4S, the system is divided into two main subsystems: on the one hand, the Node Manager (central edge management platform), and on the other, the Edge Node (industrial device deployed in the field). The Edge Node, in turn, consists of several layers: the hardware layer, which defines the physical device and its I/O resources; the infrastructure software layer (or virtualization layer), which acts as an orchestrator of the hardware resources for the applications (including the operating system and base software that manage containers); and the application layer, where the DSO's operational applications reside, usually encapsulated in containers. The Node Manager, for its part, supervises all these layers in each node and coordinates their operation at a central level.

The reason for adopting an architecture with well-defined layers is to enable massive deployments of edge nodes while ensuring that each layer fulfills a specific function and that the components at each level can interoperate and be interchangeable without vendor dependency. In this way, the aim is to avoid vendor lock-in and promote standardization and competitiveness in the devices, which in the long term reduces costs. To achieve this, E4S identifies a set of 7 standardized interfaces between the layers and subsystems, numbered from #1 to #7. Each interface specifies the communications or interactions between adjacent components, promoting these connection points as open and common for any implementation.

The project first carried out a detailed analysis of the E4S interfaces defined in the original proposal, evaluating their purpose and importance within the operation of edge computing. Subsequently, adjustments and extensions were proposed to improve the architecture based on the experience gained from the implementation of the VDC, such as clearly separating the operating system layer from the node agent layer (local management software) within the "infrastructure layer" of the edge node.

E4S initially grouped the OS and the agent in the same layer, but the design proposed in this work separates them into distinct sequential layers (OS below, Node Agent above), which improves clarity and allows for a better definition of responsibilities and interfaces for each. This separation effectively introduces a new interface between the OS and the node agent (proposed as Interface #6 in our diagram), which would cover low-level interaction.

In addition, the definition of Interface #7 and Interface #8 (not yet specified by E4S) was explored.

Illustration 2 shows the proposed model, including the defined E4S interfaces and the new or modified ones. In summary, this modular architecture enables a DSO to change the “concentrator agent” (VDC) or add other containers (e.g., a low-voltage network control agent) without changing the underlying platform, as long as standard interfaces are respected. This enables multiple applications to coexist on the same edge node and allows different manufacturers to provide interchangeable components (for example, a VDC from another vendor) in compliance with interoperability standards.

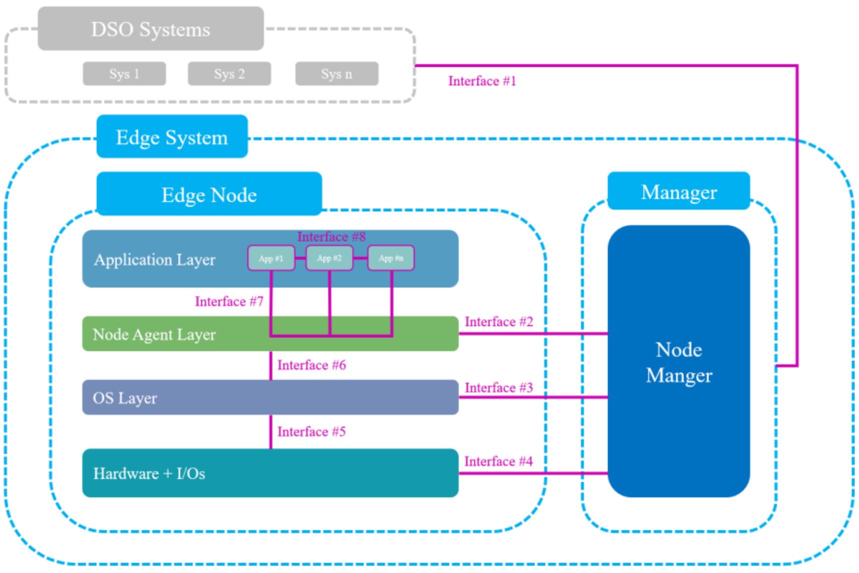


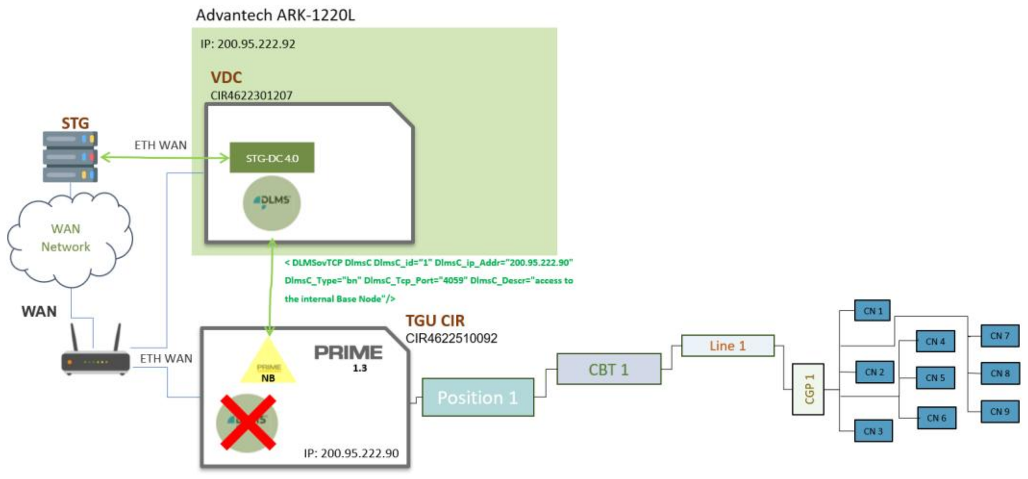
Illustration 2: Schematic of the architecture proposed in this project.

3.4 Functional Validation, OLTC Use Case, and Scalability Tests

Once the infrastructure was deployed (Barbara on-premise platform and edge nodes with VDC), its operation was thoroughly validated through laboratory tests and field pilots. Three validation areas were considered.The first consisted of functional laboratory tests of the VDC;The second involved the development of an advanced OLTC use case implemented in a real secondary substation;And the third was a scalability test by connecting the VDC to multiple field substations simultaneously.

The following sections describe these tests in detail, along with the connection architecture used in each case.

Functional tests in the laboratory: The initial objective was to verify that the virtual VDC can perform all the basic functions of a traditional physical concentrator. To this end, an edge node with the VDC was connected to an i-DE laboratory environment composed of a PRIME base node (a physical Circutor concentrator used only as a PLC modem) and several smart meters. Illustration 3 shows this configuration: the VDC (on the edge node) communicated via Ethernet with the base node, and in turn with 9 low-voltage meters installed in the laboratory, while the Barbara Panel in the DMZ managed the node.



```
< DLMSovTCP DlmsC DlmsC_id="1" DlmsC_ip_Addr="200.95.222.90"
DlmsC_Type="bn" DlmsC_Tcp_Port="4059" DlmsC_Descr="access to
the internal Base Node"/>
```

Illustration 3: Schematic of the VDC connections within Iberdrola i-DE's laboratory environment.

A sequence of three tests was defined:

1. Connectivity and configuration test: consisting of requesting an S12 report from the VDC via the central system to verify its factory configuration, and then sending a B07 command for initial parameterization (setting the IPs of the STG and FTP, as would be done with a new concentrator). Prior to this, a B07 was sent to the physical concentrator to disable its internal service and use it as a simple base node.

2. Measurement reading test: consisting of verifying that the VDC can periodically read data from meters. Reading commands were simulated (e.g., S24/G24 for device discovery on the PRIME network) and measurement queries (S21 reports for load profiles) to confirm that the VDC receives data from the meters through the base node and forwards it to the remote management system (STG) or stores it in FTP, as appropriate.

3. Scheduled tasks test: consisting of verifying that the VDC autonomously executes periodic tasks (as a physical concentrator would when collecting daily readings, billing closures, etc.). Typical tasks were scheduled, and it was verified

through event logs that the VDC executed them on time and correctly interacted with the meters.

OLTC field use case: To demonstrate the potential of running advanced applications at the edge, an On-Load Tap Changer (OLTC) voltage regulation use case was developed on a smart transformer. The test was carried out in a real Secondary Substation in northern Spain equipped with a transformer with OLTC. In this substation, the existing physical concentrator (a ZIV) was used as a PLC base node, its concentrator functionality was disabled, and the laboratory VDC was remotely connected to the base node of that substation. Illustration 4 shows this architecture, where the edge node with the VDC remained in i-DE’s laboratory (Madrid), communicating through i-DE’s network with the remote substation. The purpose of this first scenario was to implement locally at the edge an algorithm that made tap change decisions on the transformer based on voltage measurements from the meters, without relying on the cloud or human intervention, achieving near real-time control.

The MV/LV substation used for the development of the use case had 175 meters, from which a preselection of 5 representative meters was made for monitoring. The selection criteria included: choosing only three-phase meters (400 V) since they are associated with larger loads or act as supervisors of several single-phase meters; selecting at least one meter from each of the 5 LV outputs of the transformer to cover the entire area; excluding meters with unreliable communication; and prioritizing those supplying many users. Taking these selection criteria into account, the 5 optimal devices were identified, and a containerized Python application was developed on the edge node to implement the OLTC control logic. This application was designed to periodically request S21 load profile readings from the VDC for the five meters (simulating a periodic order from the central system), obtain the measured phase voltages, and calculate the average three-phase voltage of the monitored points. If the average deviates from the nominal value (230 V) beyond a tolerance band of ±3%, the algorithm decides a tap adjustment by determining whether the tap should be increased, maintained, or decreased. This decision is recorded as a tap up/down/no-change command. The algorithm runs every 5 seconds, and each cycle stores both the voltage values of each meter and the calculated tap command in a time-series database (InfluxDB) installed on the edge node. In parallel, a container with Grafana was deployed on the edge, which queries the Influx database and visualizes in real time the voltages and tap decisions. Illustration 4 schematically shows how the virtual VDC reads data from the meters (via base node), the control script processes these readings and decides the OLTC action, the data is stored in InfluxDB, and Grafana displays the voltage trends and tap commands—all within the edge node. This distributed architecture enables real-time monitoring and control directly at the MV/LV substation, without relying on the latency of communicating each reading to the central system.

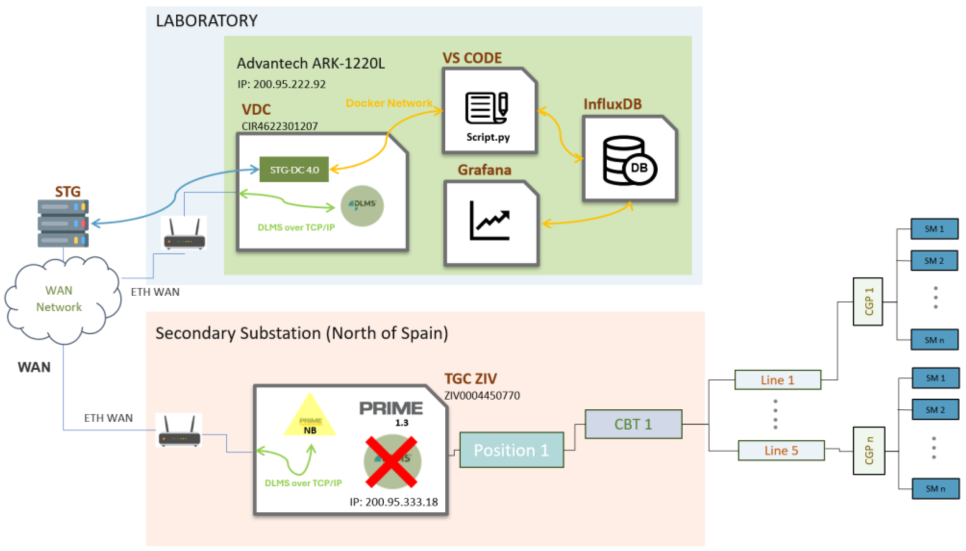


Illustration 4: Schematic of the physical and logical connections between the VDC and the SS and the edge applications.

Scalability test: In this scenario, the VDC's ability to manage multiple substations in parallel was evaluated, pushing it to its theoretical limit. In the test, 12 real secondary substations from the urban area of Madrid were connected, selected among those with the highest number of subscribers (~900 meters each). All these MV/LV substations were simultaneously linked to the VDC instance running on an edge node in the laboratory. Illustration 5 represents this configuration, where the VDC acts as a multi-substation virtual concentrator.

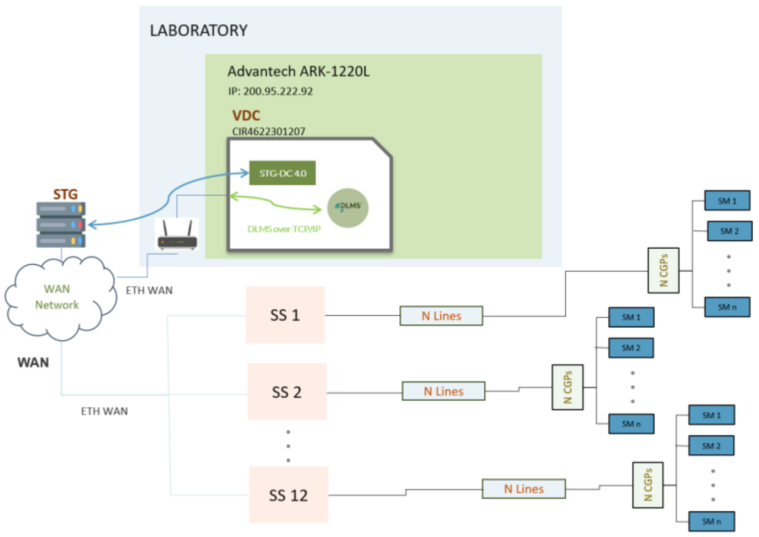


Illustration 5: Connection diagram for the VDC scalability test.

4. Results

4.1 Functional validation of the VDC

Functional validation tests confirmed that the VDC deployed on the edge node correctly reproduces all the essential functions of a physical concentrator, with minor exceptions that are solvable. Summary Table 1 summarizes the main tests carried out and their results.

Test Category	***Test***	***Executed Successfully***	***Received/Stored in STG or FTP***
Configuration & Connectivity	S12 – Factory Configuration Report	✓	✗ *(Missing fields: ipCom2, ipMask2)*
	B07 – Disable Internal DLMS	✓	–
	B07 – Initial Configuration	✓	–
Functional Measurements	S24/G24 – Meter Discovery	✓	✓
	S21 – Meter 1 (No Load)	✓	✓
	S21 – Meter 7 (Load: 60 W)	✓	✓
Scheduled Task Validation	S05 – Daily Billing Report	✓	✗ *(STG development issue)*
	S02 – Hourly Load Profile	✓	✓
	S04 – Monthly Billing Closure	✓	✓ *(STG development issue)*
	S09 – Meter Event Log	✓	✓
	T01 – Reboot Order	✓	✓
	T07 – Forced Time Sync	✓	✓
	S17 – Concentrator Event Log	✓	✓
	S24/S11 – SM List and BN Status	✓	✗ (S11 only: No BN)
	S14 – Voltage and Current Profile	✓	✗ (no LVS)

	G01 – Communication Statistics (Hourly)	✓	✓
	G02 – Communication Statistics (Daily)	✓	✓
	G12 – DC Performance History	✓	✓

Summary Table 1: Summary of the VDC validation test results.

In summary, the VDC was able to receive and respond to configuration commands (S12) and parameterization orders (B07), discover meters on the PRIME network (S24/G24), query hourly measurements (S21), and enter data into the central system, as well as execute scheduled daily, hourly, and occasional tasks autonomously. All test cases yielded satisfactory results in terms of execution in the VDC, and in most of them the report reached the remote management or target storage system as expected. For example, it was verified that the VDC, after an S24 order, correctly detected all meters connected to the base node, and that in response to S21 reading queries, the values were stored on the FTP server. Likewise, in periodic tasks it was observed that the VDC promptly initiated each request and transmitted the responses from the meters.

Some minor incidents were identified during these tests, but they were not attributable to VDC failures, rather to limitations of the test environment. For example, the initial S12 report was rejected by the RMS due to missing mandatory fields (ipCom2, ipMask2) in the configuration, which could be easily resolved by adjusting the specification so that these fields are not required or by filling them with default values. Also, in the scheduled tasks, certain reports such as S05 and S17 were not stored in the development STG due to known issues in that environment, which would not occur in the updated production environment. In any case, no fundamental technical impediments were found; the VDC proved to be technically viable as a replacement for physical concentrators, requiring only minor adaptations to the existing specification. This is a crucial result, as it clears doubts about the basic functionality of the VDC within the i-DE infrastructure. In conclusion, the laboratory tests validated the feasibility of deploying VDCs as replacements for physical DCs, fulfilling the expected functionalities once configuration details are resolved.

4.2 OLTC Use Case Results

In the OLTC pilot, the VDC's ability to enable real-time control at the edge was evaluated. During the field test (CT with i-Trafo), the control algorithm at the edge node was active for several hours, continuously monitoring the voltages of the 5 selected meters and making decisions regarding the transformer's tap. Through the Grafana interface, both the three-phase voltages measured at each point and the tap adjustment decision calculated in each iteration were observed in real time. Illustration 6 shows how initially the transformer was in the nominal tap position, with voltages within range. Then, in response to a voltage increase in certain phases beyond the 3% tolerance, the algorithm detected the deviation and began issuing commands to lower the tap in order to reduce the supplied voltage. The graphs show how, after the upper threshold was exceeded, the algorithm repeatedly recommended lowering the tap by one step.

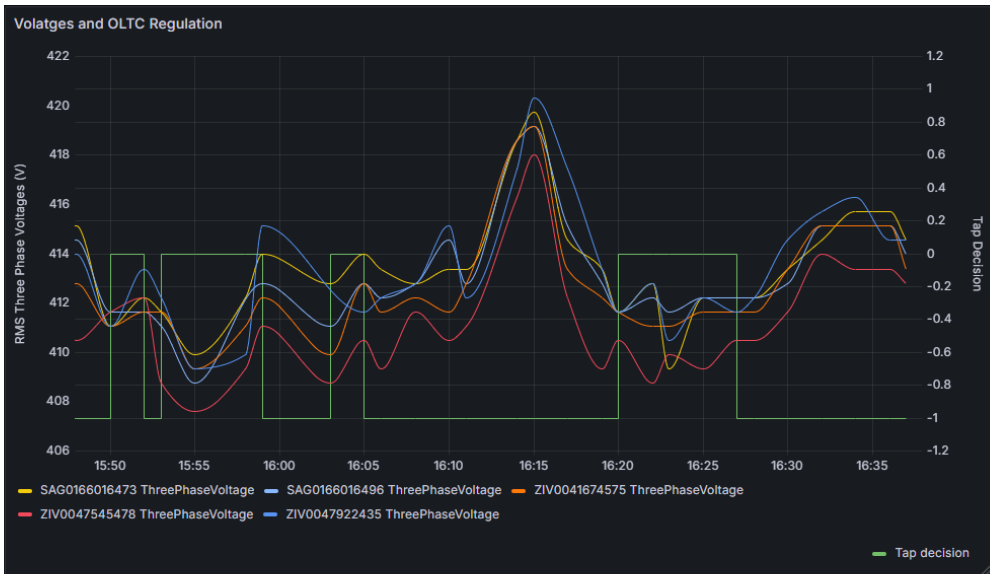


Illustration 6: Tap decision and three-phase voltage profiles.

It is important to note that, for operational and safety reasons, the commands were not physically applied to the transformer's actual OLTC during the pilot. That is, the system did not execute the tap change on the device, but only simulated it by recording it. Due to this lack of closure in the control loop, the measured voltages remained high and the algorithm continued issuing the same recommendation in each cycle. Despite this, the pilot demonstrates that the VDC and the edge application are capable of obtaining real-time measurements from the meters, processing them locally with a custom algorithm, and generating control actions with very low latency. The entire process occurred at the edge node, confirming the feasibility of implementing distributed control in the CT using the VDC infrastructure. The results visualized in Grafana showed how the system would dynamically react to voltage conditions. This use case sets a precedent for advanced applications at the edge, which can be added on top of the modular platform.

4.3 Scalability Results

The scalability test confirmed that the proposed solution can handle a volume of devices far greater than that of current individual concentrator deployments. The virtual VDC managed to maintain simultaneous communication with ~6000 meters distributed across 12 different MV/LV substations, which is equivalent to concentrating the work of 12 physical concentrators into a single edge node. This result suggests that in the future it may be possible to centralize several substations by zones or groupings into a single powerful edge node, reducing the number of devices needed in the field.

One finding was that the practical limit during the test was determined by meter security issues, not by VDC saturation. The resource usage statistics shown in Illustration 7 show that the edge node still had headroom in CPU, memory, and network while handling ~5900 connections. Moreover, although data could only be obtained from unsecured meters, the VDC did send requests to all meters. That is, the software scaled up to that number of sessions without crashing or collapsing the network. Under real conditions (with an STG capable of managing the keys), it would be expected that the VDC would successfully serve all meters. In fact, it was verified that some broadcast tasks (such as a general S05 command) were sent but failed only due to the "active key pending" issue in many of them. This demonstrates the robustness of the VDC in scheduling and handling multiple threads of concurrent requests.

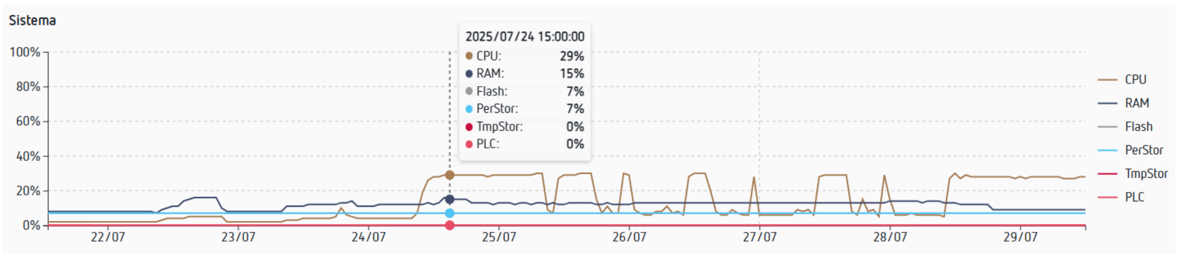


Illustration 7:Resource consumption metrics of the VDC application during scalability tests, showing CPU, RAM, Flash, PerStor, TmpStor, and PLC usage over time.

4.4 Economic Impact

The economic analysis carried out aims to compare two deployment scenarios for the modernization of Iberdrola i-DE's transformer centers (CT): the traditional scenario with the deployment of new physical TGUC concentrators versus the innovative scenario with edge nodes running a virtual concentrator (VDC). For this comparison, the initial investment (CAPEX) and operational costs (OPEX) assumptions shown in the corresponding tables, Summary Table 2 and Summary Table 3, have been used.

Device	***Cost***	***Unit***
TGUC	450.00	€/Un
LVS	70.00	€/Un
Edge Node	300.00	€/Un
NB PRIME	100.00	€/Un
License VDC	250.00	€/Instance
License DPP	175.00	€/node
Equipment installation in SS	170.00	€/SS
DPP Deployment	445,000.00	€

Summary Table 2: Capex

Concept	*Value*	*Unit*
Intervention in MV/LV Substation	140.00	€/SS
O&M interventions in SS	3%	% of SS per year
Reduction in Edge interventions	30%	% of interventions per year

Summary Table 3: Opex

Considering the total number of affected MV/LV substations, specified in Summary Table 4, a gradual and realistic deployment has been assumed, described by a Weibull-type curve (Illustration 8). Based on these assumptions, a detailed cost-benefit analysis has been carried out for each scenario over the project's time horizon (15 years), the results of which can be visually observed in the corresponding graphs (Illustration 9 and Illustration 10).

Classification	*SSs*	*SS with 2 or more positions*	*Positions MT/LV*
Total	102,054.00	22,314.00	124,368.00
Urban	55,819.00	22,314.00	78,133.00
Rural	26,185.00	0.00	26,185.00
Scattered rural areas	20,050.00	0.00	20,050.00

Summary Table 4: Number of SSs depending on their location and number of positions.

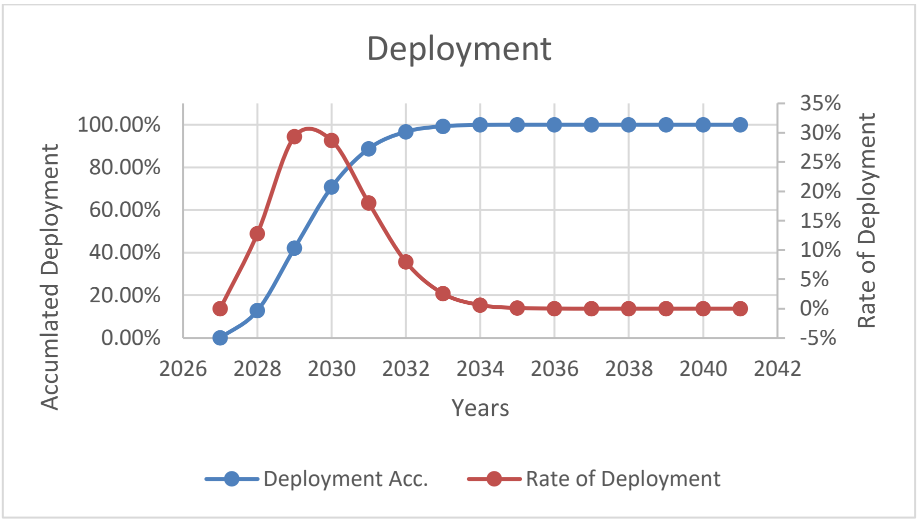


Illustration 8:Estimated deployment of Data Concentrators by i-DE for the next regulatory period.

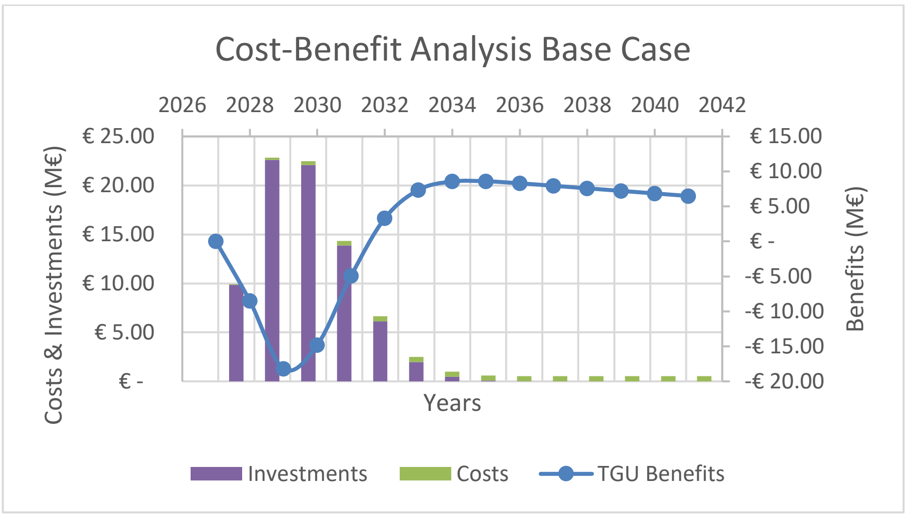


Illustration 9: Cost-Benefit Analysis Base Case.

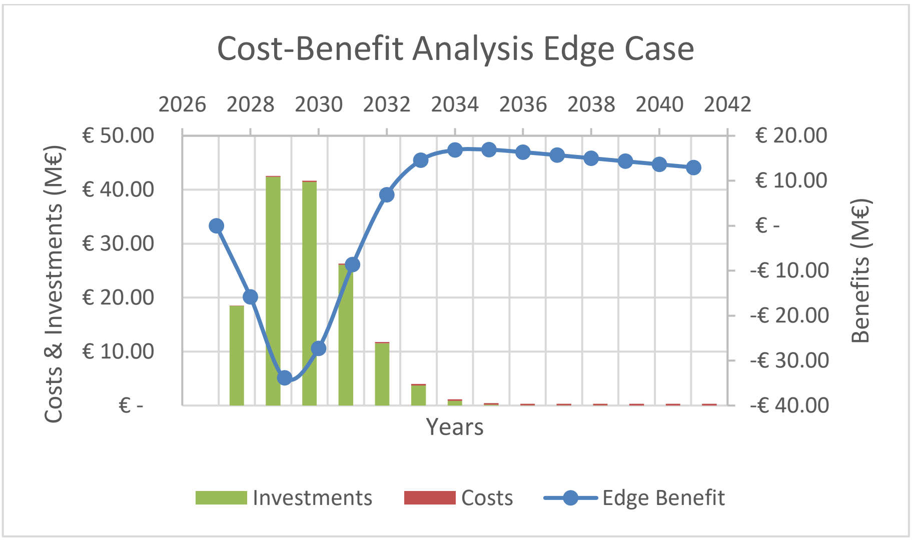


Illustration 10:Cost-Benefit Analysis Edge Case.

The key economic indicators calculated for each scenario include the Net Present Value (NPV) and the Internal Rate of Return (IRR). These indicators have been evaluated under three different assumptions regarding the regulatory remuneration of software costs: full remuneration (100%), no remuneration (0%), and the minimum remuneration required to make the deployment of edge nodes more advantageous compared to the TGUC base case. The summarized results of these calculations are presented in Summary Table 5.

C&B Analysis	***NPV***	***IRR***
Base case	5,234.11 €	6.462 %
Edge Case (50 % Remuneration)	-12,947,412.67 €	4 %
Edge Case (0 % Remuneration)	-31,080,257.63 €	-1 %
Edge Case (86 % Remuneration)	5,234.11 €	6.462 %

Summary Table 5: Summary of Cost-Benefit Analysis Results.

As the main conclusion of the economic analysis, it is worth highlighting that, given the current regulatory framework in Spain, which favors investment in physical assets (CAPEX), it has been determined that the option of edge nodes with VDC is economically more advantageous than the baseline TGUC scenario only if the regulator remunerates the software cost at 86% or higher. This threshold represents the critical point from which the edge computing alternative becomes preferable from a financial profitability perspective.

5. Conclusions

This work has demonstrated the technical feasibility of virtualizing the data concentrator (VDC) in distributed environments, validating its operation in real field scenarios. The results of the functional, scalability, and OLTC regulation use case tests confirm that the solution is capable of replicating the key functions of the physical concentrator, with additional flexibility to deploy intelligent applications at the edge. The proposed modular architecture provides a scalable and interoperable approach aligned with standardization initiatives such as E4S, facilitating the integration of distributed agents from different vendors. From an economic standpoint, the cost-benefit analysis has shown that, under certain regulatory conditions, the deployment of edge nodes with VDC can be a cost-effective alternative to traditional physical concentrators.

Taken together, the project lays a solid foundation for advancing toward smarter, virtualized networks adapted to the digital transition of the electrical system.

Table of Contents

List of Figures

List of Tables

Acronyms

AMI	Advanced Metering Infrastructure
ARE	Application Runtime Environment
BN	Base Node
CAPEX	Capital Expenditures
CIM	Common Information Model
CNMC	Comisión Nacional de Mercados y Competencias
COSEM	Companion Specification for Energy Metering
D-Bus	Desktop Bus
DC	Data Concentrator
DLMS	Device Language Message Specification
DSO	Distribution System Operator
E4S	Edge for Smart Secondary Substation Systems
FTP	File Transfer Protocol
gRPC	Remote Procedure Calls
IED	Intelligent Electronic Device
IPC	Inter-Process Communication mechanism
LDAP	Lightweight Directory Access Protocol

LVA	Low Voltage Advanced Supervisor
LVS	Low Voltage Supervisor
mTLS	Mutual Transport Layer Security
NTP	Network Time Protocol
OAMS	Operation, Administration and Managament System
OBIS	Object Identification System
OFDM	Orthogonal Frequency Division Multiplexing
OPEX	Operational Expenditures
OS	Operating System
OT	Operational Technology
PRIME	PoweRline Intelligent Metering Evolution
PS	Platform Software
RESTful	Representational State Transfer
RFI	Request For Information
RMS	Remote Management System
RTU	Remote Terminal Unit
SASCTI	Smart Automated Secondary Substation with Computing and Telecommunication Infrastructure
SEAPATH	Software Enabled Automation Platform and Artifacts Therein

SN	Secondary Node
SNMP	Simple Network Management Protocol
SOAP	Simple Object Access Protocol
SoC	System on Chip
SS	Secondary Substation
STG	Sistema de Telegestión (RMS in English)
TD	Thing Description
TGUC	Telegestión Universal Completa
TLS	Transport Layer Security
TPM	Trusted Platform Module
UDS	Unix Domain Sockets
VDC	Virtual Data Concentrator
VM	Virtual Machine
W3C	World Wide Web Consortium
WoT	Web of Things
WS	Web Services
XML	Extensible Markup Language

Chapter 1. INTRODUCTION

1.1 CONTEXT

Traditional electrical distribution network has been evolving during several years into Smart Grids, a paradigm lead by the growing need of advanced digitalization, the decarbonization goals and the increasing appearance of Distributed Energy Resources (DERs). According to Iberdrola "Smart Grids are electricity networks that can intelligently and dynamically integrate the actions of all the users connected to them – those that generate energy, those that consume energy or those that do both – in order to supply electricity efficiently, sustainably, economically and safely" [1]. Therefore, evolution towards smart grids in distribution networks aims to optimize reliability and efficiency through real-time monitoring, bidirectional power flows and adaptative control. In this context, modernization of Secondary Substations (SSs), plays a pivotal role towards this evolution, functioning as critical nodes that connect medium voltage (MV) networks with low voltage (LV) consumer endpoints. These substations, which were hardware conceived in a static and hardware bound infrastructure is now facing unprecedent challenges like integrating DERs, allowing predictive maintenance, managing dynamic grid operations or processing measurements from smart meters.

Currently, the use cases implemented in the SSs depend on proprietary hardware solutions, like Data Concentrators (DCs) or Remote Terminal Units (RTUs) to achieve essential functionalities like data aggregation, advance supervision, remote control and monitoring. This specific hardware equipment presents challenges in sourcing alternative suppliers due to the high degree of customization in both hardware and software, and the complexity of the product. Moreover, even though the current solution of i-DE presents excellent scalability, managing over 120,000 positions and approximately 11.6 million smart meters, it lacks from flexibility to adapt to innovation changes or new regulatory requirements. This is mainly due to the actual rigid development cycle, which usually involves the definition of specifications and requirements, that takes around 6 months, wait for public tendering, that usually takes between 1 and 2 years, manufacturer development, that typically takes around 1 year and finally the subsequent certification and piloting phases, that can go up to 1 year. As a result, deploying new functionalities can take up to four years. Hence, the current trend is shifting towards the use of Intelligent Electronic Devices (IEDs) as a unified platform inside a generic hardware that integrates the functionalities of DCs, RTUs, and Low Voltage Advance Supervisor (LVAS) using virtualization or containerization techniques.

Furthermore, although standards such as IEC 61850 have significantly improved interoperability in smart SSs introducing standardized communication services, data models and protocols for IEDs, the challenges and limitations described are not fully solved.

As a result, the convergence of virtualization and edge computing arises as the main solution to these problems. Edge computing decentralizes data processing, reducing latency and allowing real time responses for several applications such as load balancing, fault detection or real-time data analysis facilitating the operation of LV networks. On the other hand, virtualization decouples software and hardware allowing traditional embedded solutions to run on generic edge devices. This shift aims to reduce CAPEX by unifying different legacy devices into a single generic hardware node enhancing operational flexibility. A key initial use case for virtualization is the data concentrator, as it plays a pivotal role and provides data that is required by other applications. Therefore, virtualized data concentrators (VDCs) are particularly strategic as they can adapt to continuously evolving protocols like PRIME or DLMS/COSEM, and enable an efficient integration of new functionalities, business or costumer requirements, and regulatory demands without the need of physically upgrading thousands of nodes.

To meet with these new requirements, multiple edge orchestration platforms have been developed in order to efficiently manage containerized applications that can be deployed across different industrial environments. Solutions like Barbara IoT [3] or Onesait Phygital Edge, among other distributed processing platforms discussed in [4], have arisen to provide secure and scalable software deployment on heterogenous edge nodes. In this context, Futured, the Spanish technological platform for electric networks, formed by Iberdrola and multiple other companies related with the electric sector, has proposed the transition from the traditional secondary substation into a Smart Automated Secondary Substation with Computing and Telecommunication Infrastructure (SASCTI) as defined in [5].

Along with this architectural evolution, the electric industry is advancing towards interoperability initiatives, like the E4S alliance which aim is to define common standards and architectures that facilitate collaboration and innovation between Distributed System Operators (DSOs) and vendors, enhancing the efficiency and encouraging the use of modular and vendor agnostic solutions instead of monolithic systems. Furthermore, the exposure of edge devices to external networks brings cybersecurity challenges, particularly in communication pathways like the one between the DC and the Remote Management System (RMS), where the transition from insecure HTTP protocols to mutual TLS secured communication accredited by a Public Key Infrastructure (PKI) has become essential guarantee secure communication.

In the context of this transition, the electric distribution utility i-DE aims to investigate, test and validate the technical feasibility of virtualizing core substation functionalities such as the VDC, that emerges as a strategic solution enhancing flexibility, interoperability, scalability and possibly allowing cost-effective SSs. This solution serves as a foundational use case that will allow the development of additional applications that leverage the data collected from smart meters. Moreover, the fact of separating software agents from proprietary hardware and allowing their deployment through generic edge platforms in containers allow utilities to respond faster to continuously evolving regulatory and technological requirements. Nevertheless, this redefinition of the current framework presents several practical challenges such as ensuring compatibility with legacy systems, defining clear and common specification requirements, in order to allow migration between platforms, or securing communications. Addressing these challenges is essential to achieve true interoperability and interchangeability, and to prevent vendor lock-in.

This project aims to address the afore mentioned challenges investigating, documenting and deploying in an OT environment using the Barbara IoT platform.

1.2 *OBJECTIVES OF THE WORK*

The main objectives of this work are the following:

1) Critically analyze the Barbara IoT platform in a laboratory environment, studying its architecture, testing its capabilities to manage and orchestrate different applications and nodes and evaluating the performance of the system during real-time operations. This evaluation includes a comparative analysis of different Distributed Processing Platforms (DPPs) for edge computing to identify strengths, limitations and compliance with smart grid requirements.
2) Document the complete deployment process of the VDC on the OT environment, including the installation of Barbara OS on a generic industrial hardware.
3) Define and evaluate the requirements that a DPP must fulfill to be provider agnostic in terms of operation, focusing on interoperability, agent migration and alignment with emerging frameworks like the E4S alliance. This comprehends defining the architecture that the platform must have to allow the substitution between different software agents.
4) Validate the pilot deployment of a VDC application designed by Circutor, testing its technical compatibility with the actual smart metering infrastructure based on PRIME. This includes, establishing and testing bidirectional communication both northbound with the RMS and southbound through the base node with the smart meters, as well as developing a possible future use case for the edge computing solution in the secondary substation.
5) Study the economic feasibility of scaling up the edge computing solution as part of Iberdrola's broader grid modernization strategy and comparing it with the current solution based on the Spanish remuneration model.
6) Study the scalability options that the virtualization of the data concentrator provides to Iberdrola i-DE.

Chapter 2. TECHNOLOGIES AND RELEVANT THEORY

2.1 DATA CONCENTRATORS

A Data Concentrator (DC) acts as an intermediate field device in an Advanced Metering Infrastructure (AMI) and serves as a link between the RMS of the utility and the distributed smart meters. These DCs are installed inside the SSs, and typically, each SS hosts one DC that manages numerous smart meters connected downstream to the LV feeder. The main role of the DC is to aggregate and manage smart meter data and control commands a group of smart meters following orders received by the RMS of the utility.

Data concentrators are meant to perform numerous functions to support metering operations and remote management such as the following that can be deducted from [7]:

<u>Periodic data Acquisition</u>: The DC periodically collects metering data from each smart meter, which includes hourly load profiles, registers of energy consumption, power quality measurements and all kind of billing data. This data collection is normally done by scheduling daily reading tasks that uses standard DLMS/COSEM protocol over PRIME to communicate with the smart meters. Besides, the DC allows the temporary storage of the retrieved information in its internal database to send it to the RMS when required.

<u>Event and alarm handling:</u> The DC monitors the status of the smart meters connected to its base node controlling and managing the events and alarms generated by them or by the DC itself, such as, tampering alerts, power outage or restoration notifications and voltage quality events. These events can be captured both spontaneously and asynchronously from the meters during polling cycles. Once, the events and alarms are received by the DC, it logs and forwards them to the tele management system in near real time for further analysis.

<u>Bidirectional meter communications:</u> The DC acts as a communication hub, maintaining two-way communication with southbound smart meters over a power line communication

(PLC) network. This communication is done via a PLC modem, that is usually integrated and serves as a base node that stablish and manages the local PLC network. Through this PLC connection the DC can send commands to the meters, detect newly connected or installed devices and retrieve data from them.

Communication with the central system: The DC also provides northbound connectivity with the utility's central system using an IP network. In the past, the communication between the DC and the RMS of i-DE could be done either connecting the DC to a router that connected the DC to the RMS or directly connecting the DC to the RMS through an integrated 2G/3G modem. Currently, the devices used by i-DE do not include the 2G/3G modem, so all the connections are done via a router. Besides, depending on the nature of the request from the RMS the DC can respond with information already gathered on its database or can directly ask to the smart meters connected to it and is able to operate in both push or pull mode, that is initiating data transfer when an event occurs or answering requests from the RMS. This bidirectional communication ensures that the RMS can manage and configure the DC and the smart meters as required and can retrieve every piece of information from them.

Remote command execution: Another critical function of the DC is executing remote control commands received from the utility to manage smart meters. These commands aims to perform actions such as remote connection or disconnection of customer supply, changes in the load limit of the consumer, firmware upgrades for smart meters and the DC, time synchronization commands, or parameter reconfiguration.

Data logging and forwarding: The DC also has the capability of maintaining an internal database and logs of smart meter data or events from the concentrator itself, allowing the concentrator to periodically forward aggregated data reports to the RMS via file transfer or web services.

2.1.1 SOFTWARE ARCHITECTURE

Internally, the data concentrator software runs on an embedded operating system, usually a Linux based OS, which provides different generic capabilities such as multitask functions, device driver support or security features. The OS, as in any other device, manages hardware interfaces and gives access to services like file systems for data storage or user authentication for the device. Normally, due to the critical functions of these devices, and due to the fact exposed to physical and cyber threats, a secure OS configuration is used, minimizing most of the services to just those necessary for the DC functionalities.

On top of the OS layer, two main differentiated software modules are built on the DC in order to establish both northbound and southbound communication with the RMS and the smart meters respectively.

Regarding the module charged of managing the northbound communication, the DCs operated by i-DE communicate with the RMS following the specification defined in [8]. This document specifies how the RMS and the DC utilize Web Services (WS) with SOAP to exchange information using XML-based messages. Depending on whether the request from the RMS is synchronous or asynchronous, the message returned from the DC goes directly to the RMS or is sent to an FTP server where the RMS have access and is able to consult later.

On the other hand, a DLMS/COSEM module manages the southbound communication using OBIS codes. Basically, when the RMS sends a request to the DC, the DLMS module maps the required data parameters to the to the specified OBIS codes which are then sent to the base node that communicates with the meters injecting PRIME in the lines. Once, the information y sent back to the DC from the meters, the DLMS module aggregates the information, parses the data into XML format and sends it back to the RMS or the associated FTP server.

Additionally, the DCs usually include and embedded web server that provides a user-friendly interface for local or remote configuration. This web allows authorized engineers to access to the management console, enabling them to configure essential parameters such as the base node's IP address, the different RMSs that will communicate with the DC, allowing the connection and disconnection from the base node and facilitating the read of information such as the smart meters status in a more human way.

2.1.2 HARDWARE INTERFACES AND COMPONENTS

In terms of hardware DCs are robust industrial devices specifically designed to fulfil all the security requirements for substation environments, which may be outdoor kiosks or indoor cabinets. DCs are normally enclosed in a durable rigid plastic casing to avoid corrosion and are usually designed to allow its mounting in a DIN rail or inside a control cabinet. Besides, they are built to meet industrial temperature ratings, normally between -25 ºC and 70 ºC depending on the specifications, in order to manage the heat reached in a transformer room. Moreover, the design should be resistant to humidity and dust and must meet the electromagnetic compatibility standards, since the DC must not emit or be affected by electromagnetic interferences (The minimum degree of protection accepted by i-DE is IP2XB). Additionally, the DCs usually include status LED indicators to help the field personnel to check power connection or WAN or PLC connection.

Furthermore, the power supply of the data concentrators mut accommodate a range of nominal voltages and tolerate electrical disturbances while ensuring continuous operation. They are usually designed to support three phases input voltage and to be connected in the low voltage side of the transformer following the UNE-EN 60870-2-1 standard and must include surge protection and isolation between some of its communication ports.

On the other hand, the internal hardware of the DC must contain a dedicated CPU, along with a RAM and a non-volatile memory to store the OS and application software. The specific design and size of these internal hardware resources are tailored to handle tens of smart meters and concurrent communication sessions and depends on the requirements imposed by the electric utility to the manufacturer.

Moreover, the DC usually includes a built-in PLC communication modem, that uses PRIME technology, in the case of Spain. However, there are some models that do not include the modem internally and need an external PRIME module to stablish communication with the meters. This PRIME modems can operate as the base node, injecting PRIME on the LV network or as auxiliary nodes communicating with the base node to avoid inefficiencies as will be explained in detail in subsection Parte I2.1.32.1.3.

Besides, DCs usually present one ore more WAN interfaces to communicate with the RMS, tele management WS, FTP servers, NTP servers, LDAP or SNMP via a router or in some cases they incorporate a built-in cellular modem (2G/3G) as an alternative, but this last option is currently out of use in the case of i-DE substations. As well as a LAN interface, that enables the connection of local devices such as a LVS and allows the device to act as a router or a switch.

2.1.3 COMMUNICATION INTERFACES AND SECURITY

One of the most valuable capabilities of the DC is its ability to interface with different networks and ensure secure communication between them. These communication interfaces can be divided into two main parts, the PLC interface to communicate with smart meters (southbound communication), and the northbound interface to communicate with the RMS.

The communication with the smart meters is done via the aforementioned PLC modem that injects PRIME in the LV network. In this context, the DC can be configured as a base node (BN) or as a secondary node (SN). Ideally, in secondary substations with multiple feeder positions, such as the example shown in Figure 1, were each of them is perfectly isolated from each other, the most efficient way to communicate with the smart meters will be using each one of the DCs as BNs, since data collection will be twice faster.

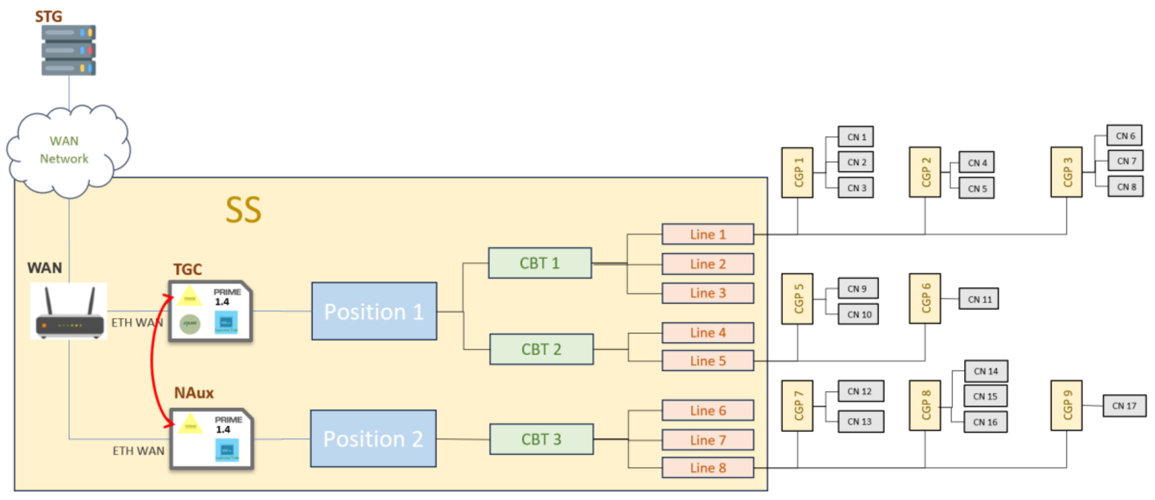


Figure 1: Architecture of a Secondary Substation with different positions.

However, it is not usual that lines from different positions are completely isolated from each other, and PRIME signals can inadvertently couple from one line to another due to electromagnetic interference, leading to duplicated meter readings and inefficient communication.

To address this issue the usual configuration when the SS manages different positions is to use one of the DCs as a BN, that is in charge of network initiation, meter registration and topology management. And the other DCs are configured as SNs, that operate passively and depend on the BN to manage meter registration and communications.

This configuration allows that the smart meters communicate exclusively with the assigned BN, that handles all communication tasks, while SNs remain inactive in terms of PRIME management avoiding repeated data processing and optimizing communication.

On the other hand, the northbound communication, is done through web services and FTP defined by the utility's RMS. For instance, the DC usually uses SOAP over HTTP to send meter data and FTP to upload asynchronous data files and has the capability to use TLS over HTTP and FTPS in order to ensure safe communication with the RMS.

In terms of security and authentication the DC is provisioned with unique digital certificates such as X.509 certificate issued by the utility's PKI, which can be checked by the RMS to avoid any rogue device to impersonating it. Besides, the DC's software includes secure storage and management of keys using a TPM and the user interface web, that it provides, requires user authentication and present different access roles which improves cybersecurity by reducing the exposed area.

Finally, these interfaces and security measures ensures that the data collected from thousands of customers is correctly and securely delivered to the RMS automizing and speeding up a lot of processes that are essential for the electric utility.

2.2 VIRTUALIZATION AND CONTAINERIZATION

2.2.1 FUNDAMENTALS AND CLASSIFICATION

Historically, modern virtualization began with VMware's ESX Server in 2001, enabling practical x86 virtualization, shortly after Lightweight OS-level isolation techniques such as BSD jails and Solaris Zones appeared, followed by Linux Containers (LXC) around 2008. In 2013 Docker revolutionized container adoption, providing developer-friendly tooling and image management and in 2015 Kubernetes emerged, becoming the standard for orchestrating containerized applications at scale, marking the transition to the cloud-native paradigm prevalent today.

Both virtualization and containerization offer the possibility of isolating the software from the hardware, but each of them present different properties that makes the suitable for different purposes.

Virtualization leverages a hypervisor to create virtual machines (VMs), which are complete software-based computers with individual operating systems and resources. These hypervisors can be classified into two different types: the bare-metal hypervisor and the hosted hypervisor. The bare-metal hypervisor, such as VMware ESXi, directly interacts with the hardware and offers efficient and secure isolation, while hosted hypervisors, like VirtualBox are suited to desktop or lab settings due to higher overhead. Basically, the bare metal hypervisor assigns resources to the virtual machines directly from the hardware and the host hypervisor shares the capabilities with a host OS which is less efficient but more suitable for desktop computers or testing.

In contrast, containerization employs OS-level virtualization by encapsulating applications and dependencies within isolated processes that only share the kernel of the host. The Linux kernel provides isolation through features such as namespaces, partitioning kernel resources, and cgroups allowing the management of CPU, memory, and I/O constraints. Besides, containers run from images built in layers that follow standards like the Open Container Initiative (OCI), allowing broad interoperability. These images are basically recipes of what the application should include, and the containers are instances build with that template or image.

The main difference between VMs and containers is that VMs provide strong isolation due to separate OS kernels but incur greater resource use and longer boot times (tens of seconds) while containers, share the host kernel, achieving lightweight resource consumption, rapid startup (milliseconds), high portability and offer slightly lower isolation levels.

Another very important concept in the context of containerization is the role played by orchestrators such as Docker Compose, Docker Swarm, or Kubernetes. These tools are responsible for managing container deployment and allow the definition of networks, volumes, and specific configurations. In particular, Docker Swarm and Kubernetes enable multi-host orchestration, that is, they automate container lifecycle management across clusters, providing sophisticated scheduling, infrastructure abstraction, and adherence to GitOps methodologies.

Besides there are also other virtualization alternatives like Micro-VMs and unikernels, that combine VM security and container efficiency.

Micro-VMs rapidly launch minimal VM instances, ideal for short-lived, secure workloads whereas unikernels compile applications into single-address-space images containing only necessary OS components, resulting in extremely fast boot times, minimal resource footprints, and reduced attack surfaces. However, unikernels present challenges such as single-process limitations, difficult debugging, and limited ecosystem maturity. These emerging technologies offer potential advantages for specialized, security-sensitive edge applications but require further development to ease industrial adoption.

2.2.2 COMPARATIVE ANALYSIS FOR CURRENT SOLUTION

For the edge node of this project, Docker Compose was selected over complex clustering solutions such as Kubernetes because it offers operational simplicity and a low learning curve, enabling engineers unfamiliar with advanced cloud-native tools to manage application stacks effortlessly.

Moreover, Docker compose ensures lightweight deployment and efficient resource utilization suitable for resource-constrained edge devices like substation computers, by sharing the host OS kernel, avoiding the overhead of separate OS instances, thereby reducing CPU and memory usage significantly compared to traditional virtualization techniques.

Besides, docker presents a mature ecosystem that brings advantages such as extensive base image libraries, streamlined integration into existing Continuous Integration and Continuous Deployment workflows (CI/CD), and widespread community support.

On the other hand, Kubernetes requires continuous management, incurs high resource overhead, and its complexity in networking and scheduling makes it unsuitable for current needs. Its alternative, Docker Swarm, is simpler but still introduces unnecessary clustering complexity which makes docker the perfect candidate to balance simplicity, performance, and reliability for this pilot phase.

2.3 INDUSTRIAL COMMUNICATION PROTOCOLS

2.3.1 PRIME 1.4 : NARROWBAND OFDM POWER-LINE COMMUNICATION

Power Intelligent Metering Evolution (PRIME) is a narrowband OFDM-based power-line communication protocol for Advanced Metering Infrastructure (AMI) networks, that enables data exchange over low-voltage distribution lines and allows utilities like i-DE to stablish communication between the data concentrator and the smart meters without the need of building new communication links. This protocol has been adopted as an international standard and is maintained by the PRIME Alliance, which is formed by utilities like Iberdrola i-DE among others, meter manufacturers, IT companies and chip providers. In order to summarize its design it could be said that it is formed by two different planes, the first one known as control and data plane follows a three layered architecture including a physical layer (PHY), a media access control layer (MAC) and a convergence layer and the second one is the management plane that enables local or remote control entity to perform actions on a node [9].

A brief explanation of the functioning of each layer based on its specification [9] is presented below:

The physical layer uses an OFDM modulation scheme where frequencies from 42 kHz up to 500 kHz can be used depending on local regulations. This range of frequencies can be divided into eight channels that could be use independently used or combined to host 97 subcarriers each that transmit the signal information. Besides these channels are separated by a guard interval of 15 subcarriers to avoid interference. Moreover, PRIME 1.4 uses differential modulation, particularly DBPSK, DQPSK or D8PSK and offers the possibility of using convolutional coding, bit interleaving and repetition code to improve reliability in noisy channels.

The MAC layer is in charge of coordinating connection an access to the shared power line between different devices. The PRIME specification provides a detailed explanation of this layer but within the context of this project it is sufficient to understand the structure of the network that this layer specifies.

The network is structured as a tree scheme as show in Figure 2, where there are two types of nodes: the base node and the service node.

The base node is the root or master node and provides connectivity to all the nodes, managing the resources and connections. This base node is unique and serves as reference for the rest of the nodes (services nodes) to register themselves in the network.

The services nodes act as branches or leaves of the tree scheme and have two main functionalities. They can act as terminals receiving or sending its own data or as switch propagating other nodes data to ensure connectivity.

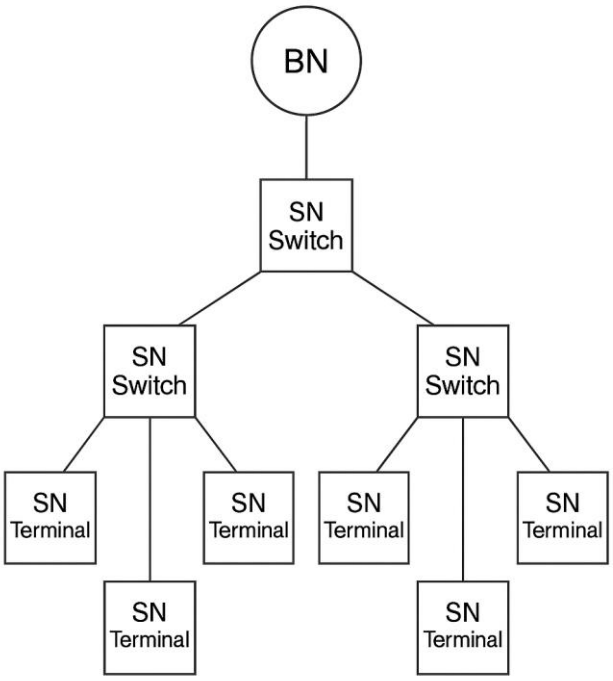


Figure 2: Scheme modeling PRIME network.

On the other hand, the convergence layer is in charge of translating high-level protocols such as IPv4, IPv6 or DLMS/COSEM to the MAC layer allowing the PRIME network to be interoperable and integrable in IT systems without protocol conversions.

2.3.2 DLMS/COSEM over TCP/IP

DLMS/COSEM is an international standard for smart meter communication that combines the protocol Device Language Message Specification, which defines how messages are interchanged between devices, and the with the Companion Specification for Energy Metering data model, which defines how objects are represented. This standard provides an object-oriented interface that allow smart meters to send data as COSEM objects identified by OBIS codes to the PRIME base node.

The DLMS application layer defines the semantics and structure of the information exchanged between a client, for example a data concentrator and a server like a smart meter. This structure provides three main types of messaging services: GET, SET and ACTION. The GET service allows the client to read different objects defined by COSEM from the server such as measurement values or configuration parameters. The SET service allows the client to modify parameters from the server such as tariff updates in the case of smart meters. And the ACTION service allows the client to request specific methods from the server like remote disconnection.

DLMS/COSEM supports multiple communication profiles over various transports, from serial links (HDLC) to network stacks (IPv4/IPv6). For example, Iberdrola i-DE uses DLMS over TCP/IP to connect the Data Concentrator with the PRIME BN. Particularly, the VDC that lacks from integrated BN, uses DLMS over TCP/IP using the Ethernet network of the SS to communicate with the external BN. This message sent using DLMS over TCP are encapsulated following the "Wrapper" profile defined by the DLMS user association that specifies a lightweight framing format, which adds an 8-byte header, containing length and port identifiers, to each DLMS Protocol Data Unit (PDU) and relies on TCP for packet ordering and error detection. This minimization of the overhead enables efficient transport of metering data through the network.

To stablish communication between the client and the server an association handshake is done by exchanging AARQ and AARE messages to authenticate and negotiate session parameters. Once the association is completed the client is ready to send request to read or write COSEM objects and receive responses.

In terms of Interoperability, DLMS/COSEM over TCP/IP enables the integration of meters into enterprise and internet networks where multiple vendors' devices adhering to the DLMS/COSEM standard can communicate with a common head-end system over IP, since the data formats and services are standardized.

In terms of Security the application layer supports authentication mechanisms (with several security levels defined by the DLMS UA) and optional encryption of messages, such as AES-based algorithms, which ensures that meter data is protected against unauthorized access or tampering. These measures, combined with IP-based network security, such as VPNs, TLS or firewalls, allow utilities to confidently transmit sensitive consumption and control data over public or private networks.

In summary, DLMS/COSEM over TCP/IP is widely employed in Advanced Metering Infrastructure (AMI) deployments and many smart electricity meters now include cellular or Ethernet modules running the DLMS/TCP/IP profile, enabling direct two-way communication with utility RMS. This setup allows remote meter reading, on-demand data collection, firmware updates, and remote connect/disconnect operations in near real-time.

2.3.3 MQTT : LIGHTWEIGHT PUBLISH/SUBSCRIBE MESSAGING

MQTT (Message Queuing Telemetry Transport) is a lightweight publish/subscribe protocol widely used for communication between IoT devices and edge computing applications. It presents an architecture that includes a broker that mediates between clients, that either publish messages to a topic or subscribe to topics to receive messages. The central broker routes the messages accordingly decoupling both clients and therefore minimizing dependencies between the source of the data and the destination. Moreover, MQTT is designed to operate efficiently on resource-constrained devices and unreliable networks, since it employs a minimal packet header and simple commands.

For example, a lightweight broker can handle message distribution between a solar inverter to a topic "microgrid/inverter1/voltage" and any authorized subscriber, such as a monitoring system, that can receive the data published for the topic. This service can be provided in three different ways or Quality of Services, that in summary are: at most once, at least once, and exactly once. Allowing the system to balance reliability versus bandwidth/latency constraints. It also includes keep-alive mechanisms and last-will messages to handle client unexpected disconnections. Additionally, MQTT is agnostic to the payload content, which usually is JSON or binary data and contrary to traditional polling mechanisms, MQTT enables event-driven data transfer, which means that devices send data only when there is new information, reducing unnecessary network traffic. This is particularly valuable in edge computing architectures, where local MQTT brokers at SSs or control centers can aggregate data from sensors and meters, perform real-time analysis, and then forward relevant events to cloud or control systems. Furthermore, in terms of security MQTT can be used alogside transport-layer encryption (TLS) and authentication, which is crucial when carrying critical grid data over public or shared networks.

All of the above characteristics make MQTT protocol well-suited to be used smart grid IoT deployments or to communicate distributed energy resources, where numerous field devices need to send telemetry and receive control signals efficiently. For instance, many utilities and vendors have adopted MQTT for new smart grid applications, such as connecting edge nodes to their DPP.

2.3.4 SOAP: XML-BASED STRUCTURED MESSAGING

SOAP (Simple Object Access Protocol) is a messaging protocol for exchanging structured data in distributed systems that uses XML to define an extensible message envelope and a framework for remote procedure calls or message-oriented data transfer. According to the W3C, SOAP is a lightweight protocol intended for exchanging structured information in a decentralized, distributed environment, using XML technologies and remaining independent of any particular programming model [10]. The structure of A SOAP message consists of an <Envelope> containing an optional <Header>, for metadata like authentication or routing, and a mandatory <Body>, which contains the payload, that is usually defined by an XML schema. SOAP messages are typically transported over HTTP/S, although the standard allows other transport protocols if needed.

In the smart grid context, SOAP-based web services have been widely used to integrate heterogeneous applications. For instance, Iberdrola i-DE uses SOAP as communication protocol between the RMS and the data concentrator. Using XML with formal schemas (often described by WSDL, the Web Services Description Language) provides a strict contract for these integrations, so that all parties interpret the data consistently and in accordance with the standard, allowing that the RMS compares the received report from the data concentrator and automatically alerts of any error in the report by comparing it with a template.

Chapter 3. STATE OF THE ART

The electric distribution network has suffered, over the last two decades, an important evolution due to the worldwide energy transition and the increasing need of utility infrastructures digitalization. The growing penetration of DERs, such as electric vehicles, PV installations and battery storage systems, is exponentially altering the behavior of MV and LV grids as exposed in [11]. These kinds of resources, which usually function in a decentralized way and cause bidirectional power flows, struggle to align with the traditional passive design of legacy distribution networks meant to support unidirectional power flows. Within this framework, the concept of Smart Grid has arisen as the solution to improve grid monitoring, enhance control capabilities, and allow greater flexibility in terms of MV and LV grid operation. One of the key components of this transformation is the modernization of the secondary substations, which constitute the main interface between the MV and LV grids. Besides, the location of the SSs made them perfect strategic spots to become distributed intelligence nodes in the future. As explained in [5], countries like Spain and France have taken advantage of the deployment of smart meters, imposed by the Directive 2009/72/EC of the European Parliament and of the Council, to implement robust telecommunications infrastructures and deploying DCs within their SSs allowing bidirectional data flow and future scalability and interoperability. Nevertheless, the technology needed within these SSs still is highly physically fragmented, depends on specific vendor and relies on proprietary hardware solutions such as RTUs, DCs or Low Voltage Advanced Supervisors (LVAs). This hinders the adoption of new standards and the update of the equipment restricting flexibility and scalability.

Besides, [5] describes three different levels of automation that a SS can have, been the third one the maximum automation level and the one expected in an SSS. The most basic level includes protection and local signaling such as automatic fault indicators in line cells, local protection mechanisms, voltage presence detection systems and auxiliary power supplies. This level of automation is intended to ensure that the substation can autonomously respond to some basic fault conditions and provide visual feedback to field personnel. The second level is called the supervision level and includes the possibility of remotely monitoring the SS sending real time information about the switching cells or electrical measurements via IP based protocols. At this level, several systems incorporate integrated web servers and usually auxiliary power systems are present to ensure the operation of the monitoring devices under adverse conditions. Finally, the highest degree of automation includes telecontrol, which allows full remote operation of the SSs and its communication infrastructure enables rapid fault isolation, load balancing and multiple coordinated response strategies across the grid. Even though, many SSs already meet most of the requirements of the third level of automation using SCADA and DLMS based systems, they still depend on rigid proprietary architectures, requiring high CAPEX for system updates, presenting multiple interoperability constraints and lacking local intelligence, that translates in slower decision-making during faults or demand fluctuations. In order to achieve this fully digital and intelligent SSs emerging strategies like virtualization will enable to break this software and hardware dependency, allowing to substitute hardware devices like DCs, RTUs and LVASs for software equivalents deployed in a generic computing node, which benefits are studied in [12]. Moreover, the decentralization of decision-making comes with the hand of Edge Computing, reducing latency and enabling real time autonomous responses.

Therefore, virtualization is increasingly irrupting in OT environments, in particular in the electrical sector, due to the flexibility, scalability and operational resilience that it offers. Hence, numerous studies have been carried out around the idea of including virtualization in SSs and the across the distribution network.

In [13], virtualizations in SSs is deeply analyzed and its role enabling the digital transition of substations is highlighted. The authors differentiate between two main types of virtualization technologies, the hardware level virtualization, that uses Virtual Machines (VMs), and the operating system level virtualization that utilizes containers instead. The document describes the advantages of hardware level virtualization using VMs, which provide better security performance due to its significant isolation which is crucial for critical operations usually carried out in primary substations and discusses different hypervisor technologies such as VMWare ESXi, Microsoft Hyper-V and Xen highlighting their benefits in efficiently managing resources, in a secure and deterministic manner. On the other hand, the document also explores operating system level virtualization technologies such as Docker and Kubernetes and highlights its suitability for SSs where their efficiency, easy scalability and lower resource consumption is needed.

Furthermore, a comparative analysis of different virtualization platforms is studied in [14], including traditional VM architectures like QEMU/KVM and cutting-edge lightweight hypervisors like Kata containers or gVisor. The document compares the performance of both virtualization approaches, concluding that even though lightweight hypervisors like Kata and gVisor have faster startup times and demonstrate better performance in terms of memory efficiency, they present potential security concerns due to the nature of containerized environments where isolation is weaker than full virtualization approaches. As a result, the authors emphasize the importance of having a good balance between performance and security and identify lightweight hypervisors as a considerable solution where fast deployment and efficiency are essential. However, in edge computing environments, such as secondary SSs, one of the main limitations of deploying traditional VMs is not performance or security, but rather the technical constraints presented by narrowband communication channels, which difficult the downloading, monitoring and maintaining large software images. Even container-based solutions present similar challenges, such as requiring the update of the underlying operating system.

Moreover, a practical implementation and an analysis of the performance of containerized applications for protection systems within IEC 61850 based smart grid environments is addressed in [15]. The analysis explores the real-time responsiveness and determinism of virtualized IEDs, showing the results of hosting up to 400 instances of virtual IEDs in a single server without compromising performance. Therefore, the authors point out the suitability of containerization for achieving modular, scalable and cost-efficient architectures for SSs while complying with IEC 61850 communication standards, showing the potential of this solution in the distribution network.

In parallel with the increasing use of virtualization, edge computing is becoming a corner stone of modern smart grid systems due to its numerous advantages. As explained in [16], edge computing brings processing and storage nearby the origin of the data mitigating the limitations that centralized cloud solutions inherently present such as latency, bandwidth bottlenecks or the insufficient velocity of responses needed for continuously changing grid conditions. Therefore, the importance of this technology in electrical distribution networks lies in the increasing volume, the velocity required, and the variability of the data generated by smart meters, DERs and IEDs since its local processing and control of tasks allows the improvement of system responsiveness and increases the autonomy of the components.

Consequently, several studies have investigated the integration of edge computing in the context of smart grids, particularly in the architecture of formed by the SSs, the different field devices and the consumers connected to the distribution network. For instance, a thorough review of edge-cloud hybrid systems is provided in [16], where the capabilities of edge computing, complementing centralized cloud services on executing critical latency tasks at the edge of the network are highlighted. Besides, the authors propose layered architectures enabling a distributed hierarchical intelligence approach where edge nodes act as connections between local devices and centralized services.

Furthermore, [15] explores the use of edge computing for high-speed data acquisition and protection functions by demonstrating the feasibility of processing sampled value streams and GOOSE messages at the edge and emphasize that the scalability and resilience of the system are improved due to the ability to off-load this time sensitive functions from the central SCADA system.

Additionally, from a practical point of view, a proof of concept of the deployment of edge computing within SSs is presented in [17]. The study highlights the benefits of local data processing for functions such as real-time load balancing and fats decision making. Moreover, the proposed architecture of this work illustrates the use of containerized applications to modularize functionalities on the edge.

The evolution of edge computing has led to the creation of DPPs designed to orchestrate containerized edge applications or virtual machines and ensuring secure remote management of the grid, scalability and fast response to real-time operational requirements. Their implementation and standardization in the grid are crucial to provide flexibility and to bring intelligence to the edge.

A comprehensive Request for Information (RFI) analysis conducted by Iberdrola in [4] defines the essential architectural components that a DPP should have to fulfill all the technical requirements that any electrical utility needs. According to the document, the DPP should be divided in a Platform Software (PS), and Operation Administration and Management System (OAMS) Agent for the PS, an Application Runtime Environment (ARE), an ARE agent, and an OAMS for the ARE software. This RFI analysis gathered and classified some of the best commercial and open source DPPs and identified key players such as Barbara IoT, Minsait, Canonical and Linux-based platforms.

The architecture proposed by Minsait is based on the eWLCA (Edge WorkLoad Consolidation Architecture), defined by Indra in collaboration with intel in [18], with some personalized modifications. The eWLCA comprehends four different levels including edge devices, edge server, edge core and edge cloud, where the edge core tier represents the DPP charged of managing the system formed by the servers and the edge devices. This DPP was deeply analyzed, tested and validated through a proof of concept demonstrating its feasibility for i-DE's electrical distribution network in [17].

On the other hand, Barbara IoT proposes an edge-native DPP in highly distributed OT environments. According to the technical evaluation carried out in [4], Barbara IoT offers a modular runtime environment that supports the execution of containerized applications on industrial edge nodes mean while offering orchestration, easy deployment and monitoring through its centralized management system. Moreover, the platform includes a marketplace with already built in applications and with the possibility of developing personalized applications containerized via docker to add multiple edge computing capabilities and includes multiple pre integrated tools for telemetry and real time diagnosis facilitating its use in electrical distribution infrastructures.

Besides, an open-source initiative is also proposed in [4] by Savoir faire Linux based on the SEAPATH project [19] developed by the Linux Foundation. As explained in [4], the solution uses the KVM hypervisor combined with QEMU for virtualization but does not allow containerization inside the VMs neither on top of the host system. Nevertheless, SEAPATH supports hardware virtualization acceleration positioning itself as robust solution, being vendor neutral and an open-source solution which comply with the utility's needs.

Additionally, another open-source solution is proposed by Canonical in [4], with a built-in platform around Ubuntu Core and the Snap packaging System, which enables secure and modular deployment of edge applications. This solution provides a more restrictive isolation than usual containers including atomic updates and roll-back capabilities but was found immature in electrical OT environments despite its advantages.

The utilization and deployment of DPPs in SSs such as the ones presented above, has led to the necessity of achieving robust interoperability between central systems, edge devices and applications, which is difficultly obtained naturally due to the diversity of hardware, communication protocols and vendors, posing a significant challenge for the scalability and compatibility among modules on edge-based architectures. As a result, even if the aforementioned platforms present different solutions that provide multiple runtime environments and orchestration capabilities, their correct integration in OT environments is conditioned to the adoption of standards for SSs.

Therefore, the smart grid sector is increasingly prioritizing the definition and use of standards that avoids vendor lock in and facilitates communication between devices across the grid hierarchy. One of the key enablers of digital substations is IEC 61850, which offers a unified framework for communication services, data modeling and device behavior standardization within substations [2]. This standard facilitates the communication and coordination among different IEDs, allowing utilities to coordinate protection, monitoring and control systems from different manufactures. Moreover, IEC 61850 also enables its application in distributed automation scenarios due to its modular and scalable nature, such as remote monitoring of DERs.

Besides, DSOs are becoming more and more interested in initiatives that involve the standardization of architectural models which objective is to achieve interoperability and scalability within the edge computing concept. For this reason, different technology companies alongside DSOs have formed the Edge for Smart Secondary Substation Systems (E4S) Alliance, which aim is to define a modular and standardize edge computing architecture based on hardware agnostic platforms where containerized agents can be deployed independently of specific vendors to perform tasks such as data acquisition or control [6]. The E4S framework defines a clear division between the infrastructure layer, that includes edge hardware and networking, the platform services layer, including orchestration, monitoring and security and the application layer that contains local agents inside containers.

Moreover, the interoperability and standardization at the application level is also crucial in distributed edge computing environments, therefore the use of well-documented Application Programming Interfaces (APIs) and standardize protocols allow the separation between the application logic and the hardware or the OS. Particularly, RESTful APIs combined with standardize specifications like OpenAPI 3.0, perfectly define communication between microservices such as applications or devices, providing precise service descriptions and real-time messages validation and allowing the generation of the client in an automatic manner [20].

Beyond standards for substation architectures and APIs, other projects such as the Web of Things (WoT) developed by the World Wide Web Consortium, try to stablish sematic and syntactic interoperability between devices in OT environments. The key component of WoT is the Thing Description (TD), which is a JSON-LD document that describes each device, including information such as its function, communication protocols, accepted commands or type of security used enabling plug and play in industrial systems as well as reducing cost and time [21]. Even though, WoT offers a promising approach in terms of edge devices integration and aligns with initiatives like E4S specifications, it is still in an early phase of adoption in the electrical sector.

Chapter 4. PROJECT DEFINITION

4.1 MOTIVATION

As detailed in the analysis carried out in the previous chapter, the digital transformation of the electric distribution network is steadily advancing towards decentralized architectures via the adoption of edge computing, virtualization and interoperable platforms that enhance flexibility, scalability and reduces latency. However, despite significant progress in these areas, several limitations persist across the existing solutions that difficult the complete digitalization of SSs.

For instance, numerous commercial DPPs have emerged in order to orchestrate, manage and provide edge computing capabilities for electric utilities, as evaluated in [4]. These DPPs exhibit diverse features such as container orchestration, modular deployment and partial compliance with industry standards. Nevertheless, none of them offer a complete and truly vendor-agnostic solution that decouples both hardware dependencies and software execution environments. Furthermore, existing orchestration technologies are not always optimized for the specific constraints and requirements of OT environments in the distribution grid.

In this context, the present project is justified by the need to evaluate, test, validate, and document the deployment of a Virtual Data Concentrator (VDC) as the first use case within a representative OT environment. The deployment uses the Barbara platform, which is an edge computing solution originally designed for IoT/IT contexts. which has been adapted to meet the specific requirements and constraints of i-DE's OT systems. Besides, the proposed work aims to deliver a lightweight, modular, and platform-agnostic architecture based on open containerization tools, ensuring compatibility with legacy systems and laying the groundwork for a fully standardized and interoperable DPP tailored to the needs of the electric distribution sector.

4.2 METHODOLOGY

The project was conducted through a structured sequence of interrelated phases, each addressing a specific aspect of the validation of the Virtual Data Concentrator (VDC) based on the Barbara Device Provisioning Platform (DPP) within i-DE's operational technology (OT) environment. All activities were carried out under the cybersecurity and operational constraints of Iberdrola's corporate infrastructure, ensuring that the methodology could be reproduced in similar industrial contexts.

The first phase consisted of an in-depth analysis of the Barbara platform and a comparative assessment with Minsait's previously tested solution, identifying key architectural differences, orchestration capabilities, and potential implications for VDC integration. This was followed by the deployment of the edge node and the on-premise Barbara platform within Iberdrola's environment, which required adapting the original cloud-based solution to an offline configuration compliant with IEC 62443, including the installation of local services such as the container registry, the over-the-air update mechanism and the certificate authority.

The third phase focused on functional validation, verifying that the VDC replicated the critical functionalities of the embedded concentrator, such as data acquisition from smart meters via PRIME and DLMS/COSEM, scheduled task execution, parsing, and communication with the Remote Management System (RMS). Once functional equivalence was established, scalability tests were performed, progressively increasing the number of managed smart meters until exceeding 7,000 units without compromising operational integrity. In parallel, a voltage regulation use case based on On-Load Tap Changers (OLTC) was developed and deployed directly on the virtualized node to demonstrate the platform's capability to host multiple advanced automation applications.

The subsequent phase addressed interoperability, analysing the VDC architecture in light of the Edge for Smart Grids (E4S) Alliance guidelines and proposing interface definitions aimed at ensuring modularity, scalability, and cross-vendor compatibility. Finally, an economic impact assessment was carried out, evaluating the cost-effectiveness of the virtualized approach compared to conventional embedded hardware under various regulatory scenarios.

Data collection was tailored to the purpose of each phase. For test validation, the main sources were the logs generated by the edge node, accessible through the Barbara platform, and the reports produced by i-DE's development telemanagement system, which received and processed data from the VDC and enabled verification of process execution and data integrity. For phases requiring specific input datasets, such as the selection of smart meters for the OLTC use case or the economic analysis, data was obtained from Iberdrola's internal databases, including network topology records, operational parameters, and asset cost information.

This methodological framework ensured a comprehensive technical, functional, interoperability and economic evaluation of the VDC, providing robust evidence to support potential large-scale deployment in smart grid environments.

Chapter 5. PROJECT DEVELOPMENT

5.1 ANALYSIS OF THE BARBARA IOT PLATFORM VS MINSAIT

5.1.1 INTRODUCTION

This section provides a comparative analysis of the two edge computing platforms that have been tested by Iberdrola i-DE to deploy edge nodes over their SSs, the Barbara IoT platform and the Minsait/Onesait IoT Edge platform. The objective of this technical comparison is to evaluate each platform's approach in a neutral, technical manner, highlighting strengths and limitations based on documented evidence. The comparison criteria include overall architecture (platform vs. edge node responsibilities), container runtime and orchestration, intra and inter-node communication mechanisms, and security primitives.

5.1.2 TECHNICAL ARCHITECTURE

The solution from Minsait includes the Indra's Onesait Edge Hub, that is an on-premises IoT Edge Management System (EMS) coupled with distributed Edge Agents on the field devices or edge nodes. The EMS, deployed in the i-DE data center, provides a web-based management console and back-end services for remote administration of edge nodes, including software provisioning, container lifecycle management, and firmware/application updates. Besides, it maintains bidirectional control-plane connectivity with each edge node over secure channels, enabling operators to remotely monitor node status, push configurations, update containerized applications, reboot devices, and even open an SSH console through a secure TLS tunnel. In the pilot, the EMS was entirely on-premise to satisfy strict cybersecurity and network isolation requirements. The data-plane in Minsait's design can encompass both local edge processing, and if needed, direct sensor telemetry to the platform that includes a multi-protocol IoT gateway (using an EMQX broker) supporting MQTT-SN, CoAP, LwM2M and other protocols for future scenarios where sensors or remote telemetry units might send data directly to the central system. However, in the PoC deployment all field data flowed through the edge node.

In terms of communication between the EMS and the edge devices, the Onesait platform uses a secure MQTT broker for machine-to-machine communication. In fact, the EMS uses a RabbitMQ-based MQTT service for issuing commands and exchanging data with edge devices in a lightweight publish/subscribe manner. Inside the platform, a reverse proxy (Traefik) is employed to route external requests to the appropriate services, and a Git-based configuration repository and a private Docker container registry are integrated to store edge application images. These components are secured by an authentication service (OAuth2), ensuring that only authorized nodes and users interact with the system. Overall, Minsait's architecture cleanly separates the central control-plane (the Onesait EMS server) from distributed edge computing on the nodes, while providing a rich set of services shown in Figure 3 to support IoT use cases.

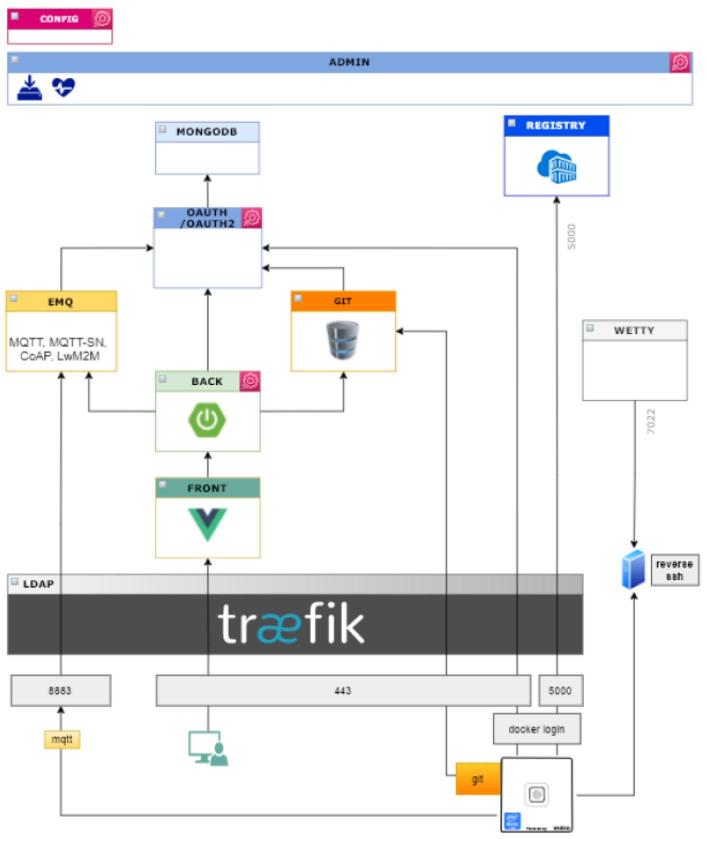


Figure 3: Onesait Edge Platform Architecture from [22]

Barbara's on-premises design is likewise fully offline and air-gapped, yet it concentrates edge-management functions in a single "Panel" that operates strictly inside Iberdrola's segmented network and exposes only the services required for device supervision and application lifecycle. The Panel publishes a management UI and API over HTTPS, provides a dedicated MQTTS endpoint that carries command and status between Panel and nodes, and serves a local container image registry and an over-the-air service for distributing operating system and application packages, all without any dependency on external networks. Name resolution is deliberately split so that user-facing hostnames and device-facing hostnames resolve within their respective planes, which allows TLS termination and certificate scope to remain correct across the DMZ and OT segments while enforcing narrowly scoped firewall rules. Internally, the Panel couples an operational datastore for device and job state, an authentication service for user and API access control, and an object store for artifacts and backups, which together keep images, configuration, logs, and metrics strictly local to the enterprise network. All exchanges are mutually authenticated and encrypted with device-specific certificates, and the enrollment flow binds a newly added node to the Panel after which the node accepts changes only from the platform, which is consistent with Iberdrola's requirement for closed-loop control in industrial. In consequence, Barbara's control plane is intentionally narrow and oriented to orchestration and supervision, while the data plane, including sensor acquisition and edge analytics, resides on the Edge Node rather than at the platform layer. As shown in Figure 4, the Panel is positioned in the industrial DMZ between the corporate IT and OT segments and exposes its HTTPS endpoints, MQTTS control channel, and local registry and artifact store to the nodes through tightly controlled routes, thereby embodying the offline, on-premises control plane described above.

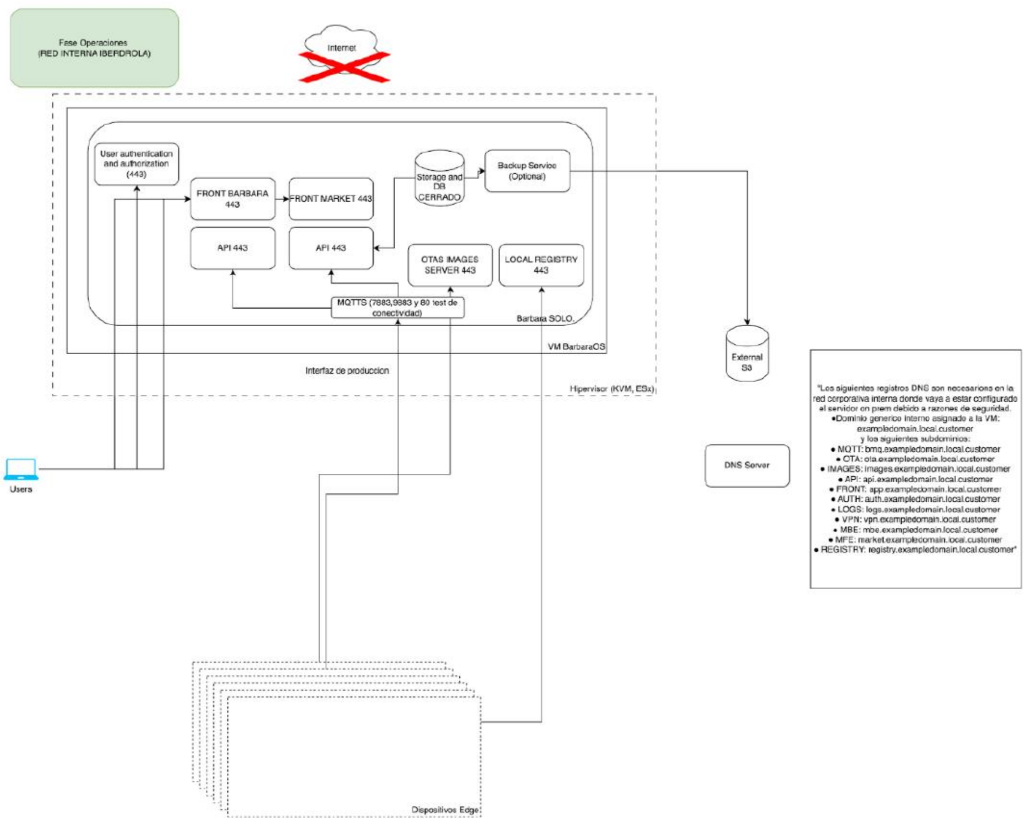

Fase Operaciones
(RED INTERNA IBERDROLA)
Internet
User authentication and authorization (443)
FRONT BARBARA 443
FRONT MARKET 443
Storage and DB CERRADO
Backup Service (Optional)
API 443
API 443
OTAS IMAGES SERVER 443
LOCAL REGISTRY 443
MQTTS (7883,9883 y 80 test de conectividad)
Barbara SOLO.
VM BarbaraOS
Interfaz de produccion
Hipervisor (KVM, ESx)
Users
External S3
DNS Server
"Los siguientes registros DNS son necesarions en la red corporativa interna donde vaya a estar configurado el servidor on prem debido a razones de seguridad.
•Dominio generico interno asignado a la VM: exampledomain.local.customer
y los siguientes subdominios:
• MQTT: bmq.exampledomain.local.customer
• OTA: ota.exampledomain.local.customer
• IMAGES: images.exampledomain.local.customer
• API: api.exampledomain.local.customer
• FRONT: app.exampledomain.local.customer
• AUTH: auth.exampledomain.local.customer
• LOGS: logs.exampledomain.local.customer
• VPN: vpn.exampledomain.local.customer
• MBE: mbe.exampledomain.local.customer
• MFE: market.exampledomain.local.customer
• REGISTRY: registry.exampledomain.local.customer"
Dispositivos Edge

Figure 4: Barbara edge platform architecture from [23].

Both platforms employ MQTT-based messaging for device-manager communication, but their scope differs. Minsait's use of an advanced EMQX broker (with MQTT-SN/CoAP/LwM2M support) facilitating direct IoT device integration with the EMS, whereas Barbara's use of an internal MQTTS broker is focused narrowly on control-plane messaging (device management commands, telemetry about node status, etc.) between the panel and the edge nodes. This design choice aligns with Barbara's philosophy that edge nodes aggregate and handle all field data internally, forwarding only high-level outcomes if needed, rather than acting as mere gateways. Moreover, security and network architecture are foundational to both solutions, Minsait's architecture leveraged corporate PKI with device certificates and TPM hardware for node authentication, and Barbara uses mutual TLS authentication for platform to node connections. Thus, at a high level, both platforms meet the same functional goal, that is to remotely manage and orchestrate distributed edge computing, but with slightly different design emphases.

5.1.3 Edge Node & Orchestration

On the edge device itself, the two platforms diverge significantly in implementation. Minsait's Edge Node software is delivered as the Onesait Edge Agent, a service that can be installed on standard Linux distributions (the pilot used Intel's Clear Linux OS, but Debian/Ubuntu or CentOS were also supported). This agent is not a full operating system but rather an application layer that interfaces with the EMS. It manages a local Docker container runtime (Docker Engine) on the node and orchestrates the deployment of edge microservices using container descriptors. The Edge Agent maintained a persistent secure connection to the EMS, through which it received deployment orders, configuration updates, and transmitted telemetry. Architecturally, Minsait's edge solution delineated system-level containers and application-level containers on each node. System containers provided common services such as an MQTT broker for internal communications, a local database for caching or buffering data, and security services, forming a foundational layer on the node. Above these, business or application containers implemented the actual use-case logic (protocol adapters for field devices like Modbus, IEC-104, or analytics algorithms, etc.), which could be added or updated independently. All containers on the node communicate through a lightweight internal message bus consisting of an MQTT broker that manages publish/subscribe messages among containers, ensuring decoupled inter-container communication. This design was explicitly chosen to facilitate inter-application interoperability, each microservice can publish data or subscribe to topics without direct coupling, and even request-response interactions can be implemented atop this bus or via RESTful calls if needed. Moreover, Minsait offered an agent agnostic from container-runtime, since the architecture anticipated support for alternatives like Podman or even lightweight Kubernetes distributions (k3s, microk8s) on the node, even though Docker was used for the first pilot. This forward-looking flexibility suggests that the Edge Agent could evolve to manage container workloads under a Kubernetes orchestrator for scaling or industry-standardization, without fundamental changes to the EMS. An schema of the edge node internal architecture is presented in

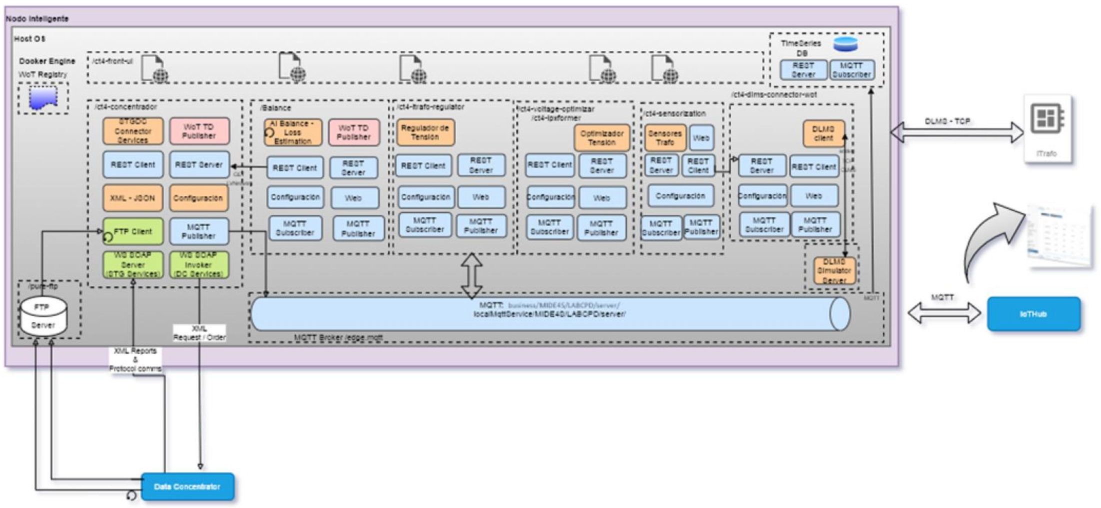


Figure 5: Minsait's edge node software architecture from [22].

In contrast, Barbara's Edge Node runs Barbara OS, a hardened Yocto-based Linux distribution that embeds the Barbara Node Manager as the local orchestrator and maintains the secure control-plane session to the Panel. The Node Manager supervises Docker containers on the device, executes deployments received from the Panel, and reports node health and application state, while updates to the operating system and applications are distributed from the on-prem registry and applied under authenticated sessions without internet access. Inter-application communication on the node is intentionally left open rather than prescribed by the platform, which allows solution architects to connect containers through direct HTTP or REST calls, shared volumes, or, when a publish–subscribe pattern is desired, by deploying a local broker as an additional container, an approach that facilitates third-party integration and accommodates heterogeneous application patterns. This openness extends to application sourcing, since vendors can package connectors and analytics as containers for publication in the enterprise registry and subsequent deployment to nodes via the Panel, thereby aligning the node runtime with a marketplace model while preserving the enterprise's control of provenance and update. Besides, operational telemetry and logs are accessible in the Panel per node and per process, including streaming views and export, and the same information is available through APIs so that Iberdrola can integrate the platform with corporate monitoring and security stacks without exposing the host operating system directly.

This differs with Minsait's more structured approach of always having an MQTT broker present. In summary, Minsait proactively addressed inter-application messaging by providing a built-in broker and defined container roles, whereas Barbara's platform leaves this design choice to the implementer's discretion. This flexibility can be powerful, but also means additional effort to achieve the same level of microservice interoperability.

5.1.4 CONCLUSION

In summary, the Barbara SOLO and Minsait/Onesait Edge platforms represent two distinct strategies to implement distributed edge computing for smart grid applications. Both solutions fulfill core requirements such as remotely managing edge nodes and deploying containerized applications within secure, offline environments, but differ notably in their approach. Minsait's proposal is closer to the E4S objectives because it delivers a prescriptive node template with a built-in inter-application message bus and documents internal structure and cybersecurity controls in depth, including PKI and TPM, tunneled remote access, a governed configuration repository, and a private registry, which together facilitate uniform interfaces and assurance at scale. Barbara, by contrast, defines a rigorous offline control plane that is self-contained and carefully segmented, while intentionally leaving intra-node communication patterns to solution design and enabling a marketplace-driven ecosystem for connectors and analytics, which encourages supplier competition and faster composition of use cases but requires Iberdrola to codify certain interoperability conventions at the application layer. Under identical on-prem and offline constraints, a pragmatic stance is to treat the Minsait architecture as a reference when uniform semantics and detailed security posture are paramount, and to leverage Barbara's flexibility when broader supplier participation and rapid composability are the priority, with the E4S interoperability work providing the formal interface contracts that reconcile both strategies..

5.2 IMPLEMENTATION OF THE EDGE COMPUTING SOLUTION IN THE OT ENVIRONMENT

The implementation of the edge computing solution in the OT environment consisted in two clearly differentiated parts. First, the physical edge nodes were installed in Iberdrola i-DE's laboratory, deploying the Barbara OS and node manager on three different industrial devices. And second, an On-Premise Barbara Platform was deployed within i-DE's network, adapting the SaaS from Barabara to an On-Premise version that meets all the stringent OT requirements imposed by i-DE. The following sections detail the process followed to implement the whole edge computing solution starting by the On-Premise platform implementation and followed by the edge hardware deployment.

5.2.1 BARBARA ON-PREMISES PLATFORM DEPLOYMENT

The initial phase of the implementation consisted of deploying the Barbara DPP within i-DE's infrastructure. This local version of Barabara's solution was carried out by Barabara's engineering team in collaboration with Iberdrola i-DE and required personalized modifications to meet i-DE's cybersecurity standards and to be able to migrate the original cloud base solution to i-DE's IT environment.

The deployed architecture was segmented into three different parts in alignment with IEC-62443 to ensure that the cybersecurity requirements of the OT environment were met. Figure 6 illustrates the different levels that include: the OT zone, where the edge nodes and all the SS devices are hosted; the demilitarized zone that hosts the virtual machine that contains the Barbara platform; and the IT zone from where users access with their PCs to the Barbara platform or edge nodes. All these zones are isolated by firewalls, with strictly communication paths ensuring minimal exposure between different networks.

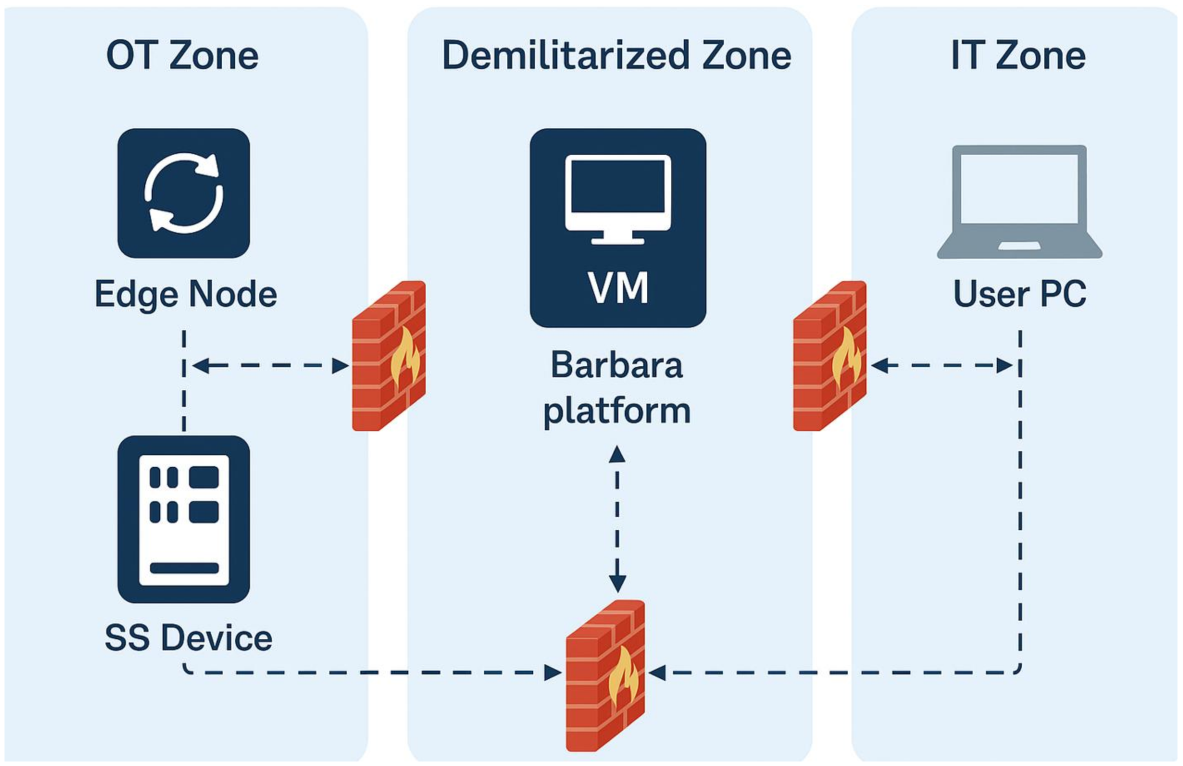


Figure 6: Schematic representation of the architecture levels of the on-premise solution.

Since the original platform was designed assuming internet connection to the cloud, the platform was completely adapted to function entirely offline. It was necessary to preload locally on the VM al the needed software, including docker container images, OS packages, update files and certificates that were securely transferred and supervised by the cybersecurity i-DE's team. For instance, components such as the docker registry, the OTA update service or the certificate authority were replicated locally to allow the correct functioning of the platform without cloud services.

Besides, secure communication was ensured, using communication channels encrypted with TLS and using VPN tunnels, between each node and the Barbara platform. They were connected via a VPN interface using predefined DNS names. For instance, the platform server was configured to feature two network interfaces, one in the OT and the other one in the DMZ, allowing DNS server to adapt its response based on the source of the request. This multi-domain setup allowed a single Panel instance to serve both zones without compromising segmentation or security.

Moreover, the monitoring and update management systems were also localized, to allow that telemetry and logs from the edge nodes could be visualized on the Barbara platform and saved locally and periodically exported to i-DE's internal data base.

In summary, the Barabara On-Premise deployment successfully replicated its original cloud based functionalities within strictly segmented environment like i-DE's. The resulted architecture enables secure communication between the user, the platform and the edge nodes offering multiple edge computing capabilities and providing full operational control of the SS without compromising cybersecurity.

5.2.2 EDGE HARDWARE IMPLEMENTATION

Once the Barbara platform was deployed and correctly functioning on i-DE's environment, the edge node deployment process started. This process consisted in installing Barbara OS (a modified Linux) onto three different industrial computers. Barabara offers different versions of its OS that are compatible with a considerable number of devices, but the selected ones where the Advantech ARK-1220L, the Advantech EI-52 IoT and the Delta E4S. These edge devices were selected to cover a wide range of hardware capabilities and to ensure that Barbara OS could operate reliably on different architectures.

The first step of the deployment consisted in installing the hardware in Iberdrola I-DE's OT laboratory, which is configured to replicate a secondary substation. This laboratory environment is deeply explained in 5.4.1.1, but in summary it provides a PRIME base node, a transformer and nine smart meters to simulate real scenarios of a SS. Hence, the edge node was connected to the BN via ethernet LAN and to a local PC for OS installation.

Once everything was correctly connected the OS image was flashed into the edge node following the vendor's standard deployment process, which automatically boots each device and initializes the system. Once the system is initialized it is necessary to manually configure the network parameters to enable the communication between the edge node, that is in the

OT environment and the DPP that is within the IT environment separated by i-DE's firewall. Once the firewall was set up to connect the edge node IP address with the virtual machine containing the Barabara Platform, the edge node starts the communication with the DPP and it is necessary to manually register the edge node in the platform using a unique ID generated by the node and shown in its GUI.

Finally, the edge nodes were recognized by the platform and added to its inventory, allowing the remote deployment of applications and the monitoring through its interface. Particularly, the tests explained in 5.4.2 were carried out.

5.3 EDGE COMPUTING SOLUTION OPTIMIZATION FOR INTEROPERABILITY

The aim of this section is to contribute to the definition of a layered architecture for edge computing and virtualization platforms that E4S seeks to standardize by proposing some changes in the defined architecture that could contribute to the full interoperability of the edge computing solution. The goal is to present an exhaustive analysis that introduces new architectural ideas and proposes and defines the necessary interfaces to standardize communication between the different layers that form the edge computing solution.

5.3.1 ANALYSIS OF THE E4S ARCHITECTURE AND INTERFACES

E4S (Edge for Smart Secondary Substations defines in [6], a layered architecture for edge computing platforms for SSs keeping modularity and interoperability as key characteristics of this platform. The E4S proposed architecture shown in Figure 7, comprehends two main subsystems, the edge node manager (the platform) and the edge node (the industrial computer). The edge node subsystem is composed by four layers, the hardware layer which aim is to define the physical computing device used, including its I/O resources; the software infrastructure layer, that aims to virtualize and abstracts these resources for use and serves as orchestrator for the application layer; the application layer hosts, the operational applications of the DSOs, such as the virtual data concentrator, in an isolated environment normally using containerization. On the other hand, the edge node manager subsystem oversees all the aforementioned layers.

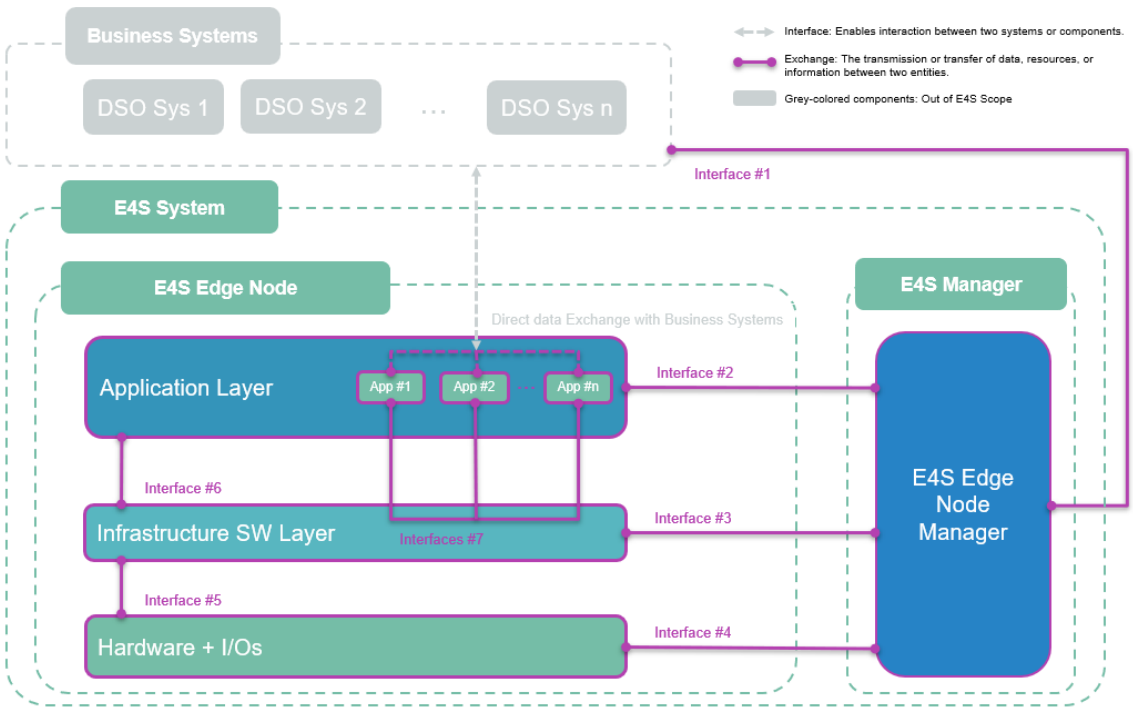


Figure 7: E4S Architecture Breakdown from [6].

The key idea behind adopting this layered architecture is to support massive edge deployment across thousands of SSs guarantying that each layer has a specific function, that all the components can interoperate smoothly and that all the layers are individually interchangeable to avoid vendor locking and promote competitivity and standardization of devices that usually leads to cost reduction.

To enable this layered approach, E4S identifies in [6], seven interfaces numbered and depicted in Figure 7. Each interface has a specific purpose and importance, an overview of their intended functions will be studied firstly, and a deeper analysis of the new proposed interfaces and the ones more related with the scope of the project will be carried out afterwards.

The first interface connects the external DSO business systems, such as SCADA or ADMS with the node manager. Its main function is to integrate the new edge platform with the existing enterprise systems, allowing DSOs to share high-level asset data, alarms or commands. Even though the definition of this interface has not the highest priority in the E4S project, Restful APIs over HTTPS could be taken into account due to is simplicity and widespread use. For instance, a REST/JSON API would allow DSOs to query information or command the Edge Nodes directly from its business systems through the node manager instead of using the administration console offered by the edge computing platform.

The second defined interface connects the node manager directly with the application layer that hosts the different applications running at the substation as the virtual data concentrator. However, the E4S questions the need of defining this interface, and this project propose to merge the function of this interface into the one that connects with the node agent layer, in order to simplify the solution and avoid redundant communications.

The third interface connects the node manager with the software infrastructure. This interface allows to define the communication protocols used to deploy or update containers, monitor containers health and send application logs as well as allowing overall configuration and monitoring of the edge node's infrastructure and firmware updates. It should be noted that the preliminary E4S definition includes the OS of the edge node and the node agent in the same layer, and this project proposes the separation of these two concepts on two different layers as will be detailed later. The definition of this interface is highly relevant for the E4S Alliance since it comprehends the main channel for OAM commands from the DSO's platform or control center to each edge node. Regarding the protocols that could be proposed for this interface, early proposals include using SNMPv3, which is commonly used for network device management and hence could be used to collect device information. Other solutions could be to use industrial standards like IEC 61850 MMS (manufacturing Message Specification) to manage data in a structured manner or to use HTTPS/REST APIs with TLS. Besides using MQTT with TLS was also an idea due to is lightness and simple functioning based on publish and subscribe to topics. Each of these options have advantages and disadvantages, for instance, SNMPv3 is secure and standardized for network management but is limited in semantic richness, REST/HTTPS is very common and simpler to read for humans, but it is not built in purpose for real time device control and MMS is powerful in substation environments but is more complex to implement.

The fourth interface connects the node manager with the hardware layer, this interface allows the DSOs to have remote access to the physical hardware of the edge node. This could include performing low-level resets, updating the firmware of physical modules, since E4S is studying the possibility of defining a modular edge node, where different interchangeable cards could be used for distinct functionalities, or out of band management of the device. Since the definition of this interface has a moderate relevance for the E4S, as will not be used in the day to day smart grid operation, it won't be detailed in the next section. But, it is worth mentioning that the E4S working group noted that an interesting possibility could be the use of DMTF DASH (Desktop and mobile Architecture for System Hardware), that is a Distributed Management Task Force standard that defines a web services interface for hardware management based on SOAP. This proposal could be promoted by the fact that many CPU/SoC vendors implement DASH. Additional proposals like IPMI or its modern successor (Redfish) which is based on RESTful APIs over HTTPs and uses JSON could be also taken into account.

The fifth interface connects the software infrastructure layer with the hardware layer. This interface is meant to define how software infrastructure composed mainly by the node agent and the OS interacts with device drivers, I/O modules or modularized hardware components. It allows to communicate relevant information like hardware identity and enables virtualization abstracting hardware from software. For this interface, the E4S alliance proposes a proprietary system management interface that consists of a small number of sideband signals and a RS485 bus that connects different modules. This interface is completely defined and already created by the E4S so there is no point to enter in more detail.

The sixth and seventh interface haven't been defined yet by the E4S alliance, but it is assumed that the sixth interface includes container interfaces, local APIs that the application might use to request services form the node agent or a way to request computing resources like CPU, RAM…, it could be seen as the application to virtualization interface. In terms of protocols some options have been listed, but not detailed like gRPC, a modern open source performance Remote Procedure Call, RestFul APIs, an IPC mechanism like D-Bus and even MQTT was mentioned but is more normally used for publish and subscribe messaging rather than direct OS services. And the seventh interface intends to define how the node agent manages applications on the node and how applications communicate with each other and discover each other services. This is one of the most critical parts and will be analysed in depth in the next section, but as a summary it defines how the platform orchestrates the DSO's apps at each substation and how each application communicates its data in a standardize way.

5.3.2 PROPOSED ARCHITECTURE AND INTERFACES

This project aims, in part, to propose an alternative layered architecture to the one initially presented by the E4S working group. This proposal inherently leads to the emergence of new interfaces, which will be addressed in this section through a preliminary technical analysis. The goal of this analysis is to provide E4S with a robust foundation to support informed decision-making regarding their edge computing platform. To achieve this, a new architecture and associated interfaces are proposed and illustrated in Figure 8.

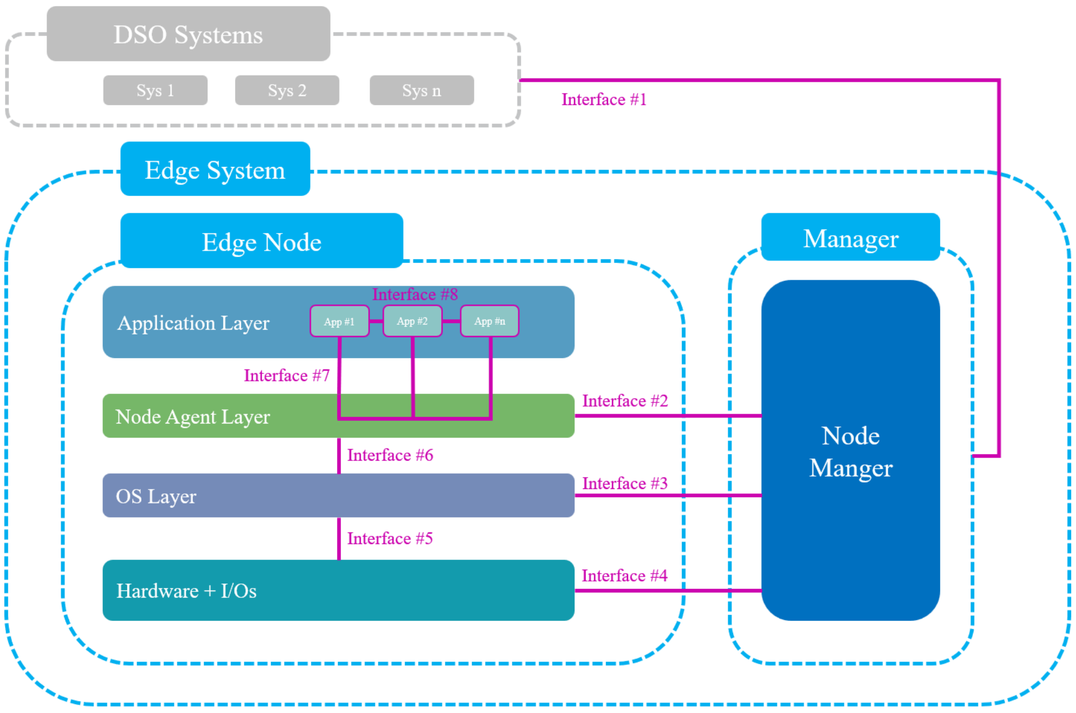


Figure 8: Proposed Edge Node Architecture and Interface Model for E4S Platform

This newly proposed architecture differs from the original architecture depicted in Figure 7 in that the single "Infrastructure Software Layer" defined by E4S has been subdivided into two distinct sequential layers, the OS and the Node Agent. This separation is intended to further modularize the edge node by enabling the Node Agent (the entity responsible for managing applications, containers, and communications) to operate entirely independently from the OS. This structural decoupling grants greater flexibility to DSOs concerning both OS selection and Node Agent providers.

Inevitably, this architectural adjustment introduces a new interface between the Operating System and the Node Agent. This newly defined interface will also encompass functionalities initially assigned to Interface #6 as specified by the E4S. Additionally, this work will explore potential approaches for defining Interface #7, focusing on communication pathways between the Node Agent and applications, and Interface #8 focusing on interactions among applications themselves.

The idea is that the new layer called Node Agent Layer takes the role of orchestrating the lifecycle and managing and communicating with the containerized applications within the edge node. Besides, this layer is in charge of managing inter-application communication and provides runtime services such as logging, local health monitoring and facilitating integration with the centralized node manager.

The following subsections present a more detailed analysis of the mentioned interfaces:

5.3.2.1 Interface #6: OS Layer - Node Agent

This interface defines the communication between the node agent and the OS in both directions. This separation intends to enhance modularity and promotes vendor agnosticism allowing DSOs to use generic open-source OS distributions while allowing the competition of different node agent's vendors.

Basically, this interface ensures the communication of telemetry data such as CPU/memory usage, device status or events, from the OS to the agent, and control messages or queries from the agent back to the OS such as configuration changes or service control commands. Since the current solutions for edge nodes are based on Linux OS, the proposed communication methods will be based on well tested protocols on Linux like IPC mechanisms native form Linux such as sockets, or message buses. Protocols such as HTTP REST APIs, gRPC or MQTT won't be considered due to its high overhead and lack of native integration with the OS kernel. An analysis of different practical approaches describing how they work and comparing their advantages and disadvantages is carried out below in order to serve as a base study to the E4S alliance.

Linux includes different possibilities to connect the node agent Layer, assuming that it is located on the user space, with the OS (kernel or user space) like the following:

The first proposed option is to use Netlink sockets, that provide an IPC that connects the kernel of Linux with the user space via the AF_NETLINK socket family, that allows to exchange structured messages between them. A Netlink socket supports bidirectional communication and provides the possibility of interchanging three different types of messages, including a "do", that performs a single action a "dump", that allows dumping information and a "Multicast", which function is to get asynchronous notifications, as described in [24]. This option presents as advantages that there is no need for protocol parsing, that is that Netlink messages are already configured to be understood by the OS, so there is no need to interpret message formats which makes it faster and more efficient, and that it is supported by Linux so it is maintained and documented for modern Linux distributions. As disadvantages it could be said that it is Linux specific, however this does not present a great problem, and customized module for the kernel may be required to add some functions, which increases complexity.

The second option is to use Unix domain sockets (UDS) that is similar, but in this case it offers a socket based IPC of the AF_UNIX family that connects the node agent hosted on the user space with a daemon that is also located in the user space. This means that Unix domain sockets do not provide direct connection to the kernel, and therefore it is suitable for interaction between system services and applications running on the user space. As a disadvantage UDSs lacks from already built-in message framing or schema, this means that it simply transmits raw byte streams, and this implies that it is necessary to define a customized schema.

The third alternative consists of using D-Bus, that is a Linux based message bus for IPC. This bus uses a daemon to route messages between different processes, in this specific case it will connect the OS with the node agent. For instance, the OS could send signal messages when an event occurs, and the node agent could call methods or listen to signals. As advantages the D-Bus messages are typed, which means that each message has a specific type, its payload could be text or binary and they can follow a predefined structure. Besides, D-Bus is well documented and standardize and supports synchronous and asynchronous signals. And as a disadvantage it has a relatively high overhead which increases latency.

To summarize the explained above, Table 1 presents a summary of the pros and cons of the proposed alternatives.

Protocol/Mechanism	***Communication Model***	***Advantages***	***Disadvantages***
Netlink socket	Local kernel to user socket (AF_NETLINK)	- Low overhead - Kernel integration - Supported by -Linux	- Custom Kernel module required
Unix domain socket	Local stream socket (AF_UNIX)	Simple IPC - High local throughput -Widely supported	- No built-in protocol (must define messages)
D-Bus	Local message bus (IPC daemon)	- Standard on Linux - Service discovery -Async/Synch signals	-Higher latency -Moderate overhead

Table 1: Advantages and disadvantages of the proposed protocols for interface #6.

In summary, each of the proposed option has trade-offs in terms of performance and complexity of integration with the Linux OS. Taking into account the aforementioned advantages and disadvantages and considering the need for efficient, low-latency, and maintainable communication between the Node Agent and the OS layer within the same industrial device, Netlink sockets appears to be the option among the ones proposed that most fit E4S requirements.

5.3.2.2 Interface #7: Node Agent – Application Layer

This interface acts as the enabler of several critical behaviors like the following:

First it enables the lifecycle management, that is that the node agent can install or remove applications, start and stop them based on remote commands or scheduled tasks and supervise their status in terms of CPU or memory usage. Second, it provides runtime services such as: logging, allowing centralizing the collection and forwarding to the node manager the application and system logs for diagnosis; configuration distribution, ensuring that applications receive the correct parameters during deployment; and local health monitoring, which continuously oversees the status of running containers. And third, it provides control over the containers, if the applications are containerized. The node agent uses container runtime APIs like Docker that allow to mount volumes or set up container networks.

All the functions described above make this interface highly relevant for DSOs, since it dictates how efficiently and reliably the applications run at the edge node. If this interface is well designed, failing applications will be restarted automatically, data will be shared securely, and local automation will be implemented in case that the connection with the node manager is lost.

Moreover, in order to propose and correctly analyze the ideal protocols that might define this interface, it is crucial to first understand the type of data exchanged through it. Some examples of data exchanged are presented below alongside their characteristics:

Application control commands: The node agent might send commands to the container runtime such as Docker, asking it to start, stop or update a specific application, which might also involve pulling a new container image or at list part of it and restarting it.

Lifecycle events: The node agent might also request a health check from the applications, for example it could request a heartbeat, that means that it can periodically request to a health check endpoint in the app via HTTP or gRPC to verify that the application responds correctly.

Configuration and registration: Normally, the node agent might need to send some configuration parameters like environment variables or secrets to the application when starting it or the application code itself. If a container orchestrator such as Docker Compose is used, the configuration defined in the docker-compose.yml file is interpreted by the client and translated into API calls sent to the Docker Engine, typically over a local Unix socket or HTTP-based protocol.

Service discovery queries: In this case is the application which can ask to the node agent to find an specific service. If for example the node agent has a registry of services like a simple naming service, it could respond to the application with the correct endpoint or the correct topic of a broker if a node agent based on MQTT is used.

Telemetry and logs: Normally the applications are designed to emit logs about their status, and interactions that it has with the user or with other applications, therefore this interface will need to transmit these logs from the application to the node agent.

Notifications or events: Besides, the interface will also interchange data of events, like app errors sent to the node agent that could then take further action.

Based on the exchanged data presented above, it is possible to identify and analyze the following protocols that could be used for this interface:

Local APIs (REST or gRPC): This approach requires that the node agent expose a local API like REST or gRPC that apps can use for configuration and management functions such as register themselves when starting or report health status. The advantage of using REST/gRPC is that they are language agnostic, structured and secure since it could be limited to localhost. Particularly gRPC is very efficient and is able to provide streaming. Nevertheless, this approach requires that the applications include a specific client for this APIs, adding complexity to the apps, and if the app crashes it could be complex to ensure it calls the API correctly. Besides, REST and gRPC are synchronous models, which may require asynchronous notifications.

IPC (D-Bus or similar): Another option could be implementing a message bus system like D-Bus which is created for inter-process communication in Linux. The idea is that the node agent publishes services on D-Bus that the applications can call and get a response, or that the applications are listening for signals that the node agent sends to the D-Bus. As an advantage this solution is very efficient for managing local messages and avoids the overhead of HTTP and is normally straightforward for processes running on the same OS. However, D-Bus usually restricts you to specific programming languages like C or Python, which constraints developers and therefore hinders efficiency. Besides, D-Bus can be very complex for large systems, and it could be difficult of debugging which is a big disadvantage taking into account that one of the most important matters for DSOs is the traceability of the problems in this modular solutions like de edge computing.

MQTT or Message Broker: This approach consists of deploying an MQTT broker with specified topics to which the node agent could subscribe for control messaging. For instance, the node agent could subscribe to topics for sending commands to the applications or receiving events from them. The idea is that both the Agent node and the apps are clients of the MQTT broker and can publish or subscribe to different topics. This option has as advantages that is robust since, it clearly decouples the application from the node agent, it is lightweight, and in case of restarting an application it just needs to reconnect with the broker. In the other hand, MQTT provides an asynchronous communication which can difficult and delay some exchange of data unnecessarily. Besides, security and access control should be managed to avoid malicious subscription to inappropriate topics and debugging MQTT control flows might be more difficult than a direct API call.

Container Orchestrators APIs: A good option in the definition of this interface, is to leverage the APIs offered by the container orchestrator. For example, if a slim Kubernetes was used their already built APIs could serve to share configuration, health status… However, even without Kubernetes, the APIs exposed by Docker could be also used and considered as a part of this interface, allowing the node agent to communicate with the Docker engine using its REST API avoiding the need of the node agent providing an API to the application. As a advantages, this solution provides simplicity, since there is no need to develop or maintain an additional API between the node agent and the applications, it also provides good decoupling between both layers and it is possible to leverage mechanisms already supported by the orchestrator. And as disadvantages, it is not possible of changing apps configuration in real time, it will be needed to restart the container, and there is no direct communication between the application and the agent, meaning that the app cannot ask for services to the node agent unless an additional MQTT broker is used.

A summary of the given options and their pros and cons is presented in Table 2.

Approach	***Description***	***Advantages***	***Disadvantages***
Local API (REST / gRPC)	The Node Agent exposes a local API to allow apps to register, send health reports, and request configurations or services.	- Language agnostic - Structured and secure	- Applications must implement specific client logic - Risk of failure if the app crashes - Synchronous model by default
IPC (D-Bus or similar)	Uses an internal Linux message bus (like D-Bus) for inter-process communication between the Node Agent and applications.	- Efficient for local messaging - Low overhead compared to HTTP - Designed for same-host processes	- Language constraints - Complex to debug and scale - Less flexible for modular or containerized architectures
MQTT	Agent and applications act as clients of a local broker and communicate via publish/subscribe on Predefined topics.	- Asynchronous and decoupled- Robust against restarts - Lightweight and scalable - Common in IoT/edge platforms	- Topic-based security and ACLs must be enforced - Debugging control flow is more complex - Less suitable for request/response communication
Container Orchestrator APIs	The Node Agent leverages the Docker or Kubernetes API to deploy, configure, and monitor applications.	- Avoids implementing custom APIs - Reuses mature and documented orchestration tools decoupling of app and agent logic	- No direct app to agent communication - Runtime configuration updates usually require container restarts- Limited-service discovery capabilities

Table 2: Description, advantages and disadvantages of the proposed protocols for interface #7.

In conclusion, the selection of a correct communication mechanism for this interface plays a pivotal role in ensuring robust and modular edge node behavior. Hence, after analyzing the proposed alternatives shown in Table 2 and based on the actual solutions for edge computing in SSs, it could be concluded that a MQTT driven design potentially complemented by container orchestration APIs could be a promising approach due to its decoupling benefits, resilience to process restarts and for it lightness and simplicity.

5.3.2.3 Interface #8: Applications

This interface is in charge of providing semantic and communication interoperability between applications in an standardize way, allowing that different applications from different vendors have the basic rules to expose their data as if they were a black box. For example, one application could be the VDC and another application could be in charge of doing power balances or to execute a connectivity algorithm based on the information of the smart meters collected by the VDC. The correct definition of this interface will allow to stablish rules on how this interactions or publication of information should be done in order to allow apps to consume data or services from each other.

The idea is similar to classic industrial middleware or service buses and since in a SS it is usual to have different applications that need to access to the same information it could be reasonable to allow them to share data through a common bus instead of letting them maintain separated communication links.

As done before, prior to the suggestion of protocols that could be used in this interface it is necessary to define the type of data exchanged expected between these applications. Since, the edge node will be used by DSOs in a secondary substation context, the expected type of data will include all kind of measurements, control commands, set points or alarms and notifications. Therefore, the data could be scalar values, time series, or complex objects. Besides, it is necessary to understand the problem that this interface aims to solve in order to propose protocols for it. For example, an application could be publishing the voltage of the phase of a line as "V1a" and another application could be looking for "VoltageLine1PhaseA". This would require that both apps agree on a model where both know the define scheme of the message payload.

Based on the type of data exchanged and the problem that the interface aims to solve, it is possible to consider two prominent approaches combining existing industrial standards and frameworks:

Sparkplug B with CIM: This approach uses Sparkplug B, that is an open-source specification based on MQTT, that standardizes the structure of the data (payloads) and the topics naming conventions to facilitate data transmission between industrial IoT platforms. Besides, Sparkplug B a binary payload format thanks to the protocol buffers system that optimizes bandwidth utilization and enables real time data reporting. In addition, Sparkplug B will be combined with CIM (Common Information Model), that is a well stablished IEC standard (IEC 61968/61970) that provides semantic definitions of electrical measurements, entities and system configuration, to enhance its application in smart grids environments.

This combination allows to exchanged data in an efficient and structured way, transmitting via clearly defined MQTT topics grid measurements, device statuses and events. This definition brings multiple advantages like the following:

Efficiency and real time performance: its binary format ensures low latency and could be suitable for real-time grid operations.

Industrial adoption: its largely existing implementation in industrial environments proves its reliability.

Simplicity and clarity: the structured topics and payload definitions make it very simple to integrate and debug.

However, it also presents disadvantages such as:

Lack of native semantic interpretation: Even though Sparkplug B payloads are structure do not inherently provide semantic self-description and therefore require prior knowledge or external documentation.

Need for prior agreement on data models: There is a need of pre-establishing conventions, which limits flexibility when integrating evolving systems.

WoT with CIM: This approach proposes the use of WoT, that is a framework developed by the W3C and which facilitates the interoperability, discoverability and understandability between devices and applications in the IoT environment. This framework focuses on describing devices using Thing Descriptions (TDs) that are JSON-LD based documents where the properties, capabilities, actions events or associated data schemas can be defined. This with the combination of CIM allows each application to present self-describing interfaces, facilitating dynamic discovery and integration. Besides, WoT it is not tighten to a single protocol, so it can be used over HTTP, MQTT, CoAP and others to interact with the thing's properties.

For example, an application that needs to transfer voltage measurements to another, could expose a TD that annotates its voltage property via a CIM class so that the other application can read it and understand it. This TD specifies how the data can be accessed for example vi an MQTT topic and the other application can discover this TD and subscribe to the topic to receive the voltage reading.

As advantages this approach includes semantic interoperability, dynamic discovery of application properties thanks to the things description, and flexibility and scalability, since self-describing interfaces facilitates adding or updating topics to which other applications can communicate. Nevertheless, this approach also presents disadvantages like an increased complexity, since a proper implementation of WoT requires expertise in semantic web technologies, additional computational load due to the introduced overhead and increased latency compared to simpler binary payloads.

A summary of the analysis of the proposed solutions for this interface including the advantages and disadvantages is presented in Table 3.

Protocol Option	*Advantages*	*Disadvantages*
Sparkplug B + CIM	• Efficient binary payload (low latency) • High industrial adoption • Simplicity of implementation and clarity in data structures	• Lacks native semantic interpretation • Requires prior agreement on data models and naming conventions
WoT + CIM	• Rich semantic interoperability (JSON-LD, TDs) • Supports dynamic discovery and integration • High flexibility and scalability	• Higher complexity in implementation (semantic web expertise) • Greater resource and operational overhead (processing, memory, latency)

Table 3: Advantages and disadvantages of the proposed protocols for interface #8

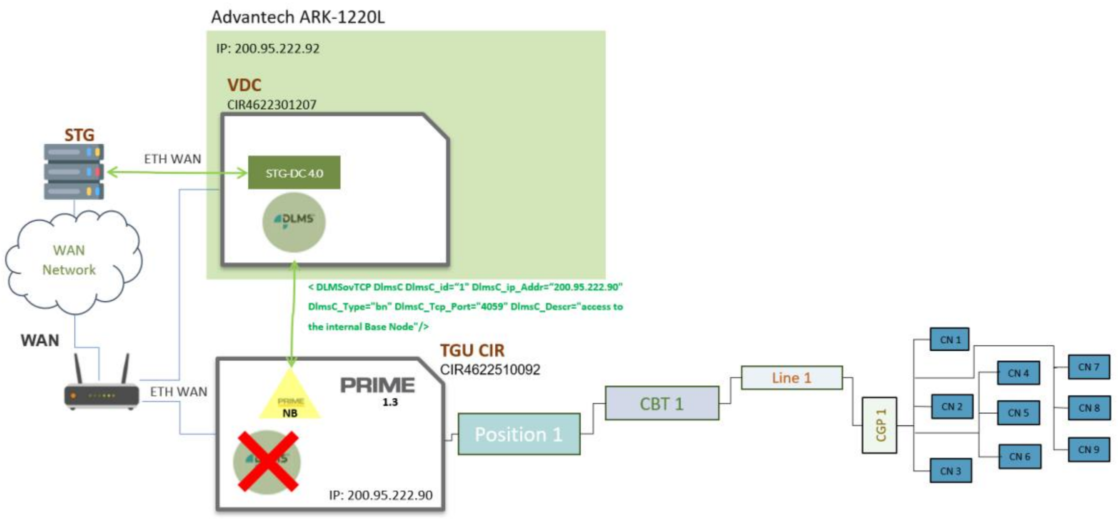
Finally, selecting between these two options involves finding the perfect balance between efficiency, simplicity and flexibility in terms of possible definition of properties, actions, measurements… On one side, Sparkplug B offers robust real-time performance and simplicity while WoT is more flexible allowing extensive semantic interoperability. Since, the applications expected to be used in a SS environment could be very complex and defined ad hoc for each DSO, it may be more likely to opt for a more flexible interface like the WoT option, as was implemented in [22].

5.4 VALIDATION OF CIRCUTOR'S VIRTUALIZED DATA CONCENTRATOR

5.4.1 PILOT SETUP AND EXPERIMENTAL ENVIRONMENTS

5.4.1.1 Laboratory pilot

The laboratory set up provides the possibility of testing the VDC or a common data concentrator from Circutor depending on how the logical connections are made. The connections and configuration for the VDC testing is shown in Figure 9, where it can be seen that the VDC application was containerized on an Advantech ARK-1220L edge computer (IP: 200.95.222.92) and has been connected via its DLMS client module using TCP/IPv4 through the port 4059 to the Base Node of the Circutor's data concentrator (IP: 200.95.222.90). In order to allow this connection, the internal DLMS over TCP interface from Circutor's DC was disabled. Besides, the scheme shows how the PRIME base node connects with nine smart meters via PRIME deployed in the same phase and mounted on a low-voltage test panel.



```
< DLMSovTCP DlmsC DlmsC_id="1" DlmsC_ip_Addr="200.95.222.90"
DlmsC_Type="bn" DlmsC_Tcp_Port="4059" DlmsC_Descr="access to
the internal Base Node"/>
```

Figure 9: Schematic of the VDC testing architecture in i-DE's laboratory

The image shown in Figure 10 shows the physical arrangement already mentioned in the schema. The edge node is not shown in the image, but it is possible to see the DC from Circutor located inside the drawer at the left-hand side of the image, and the LV panel containing the nine smart meters. Besides, it is possible to see that a load was connected to the Smart meter (CN7). The load used is a 60 W incandescent lamp, that will allow to ask for measurements and load profiles in the test environment. The image also shows a switch that allows to mirror all the traffic that flows through the router to facilitate analysis and debugging error of connections between the RMS and the VDC or the DC.

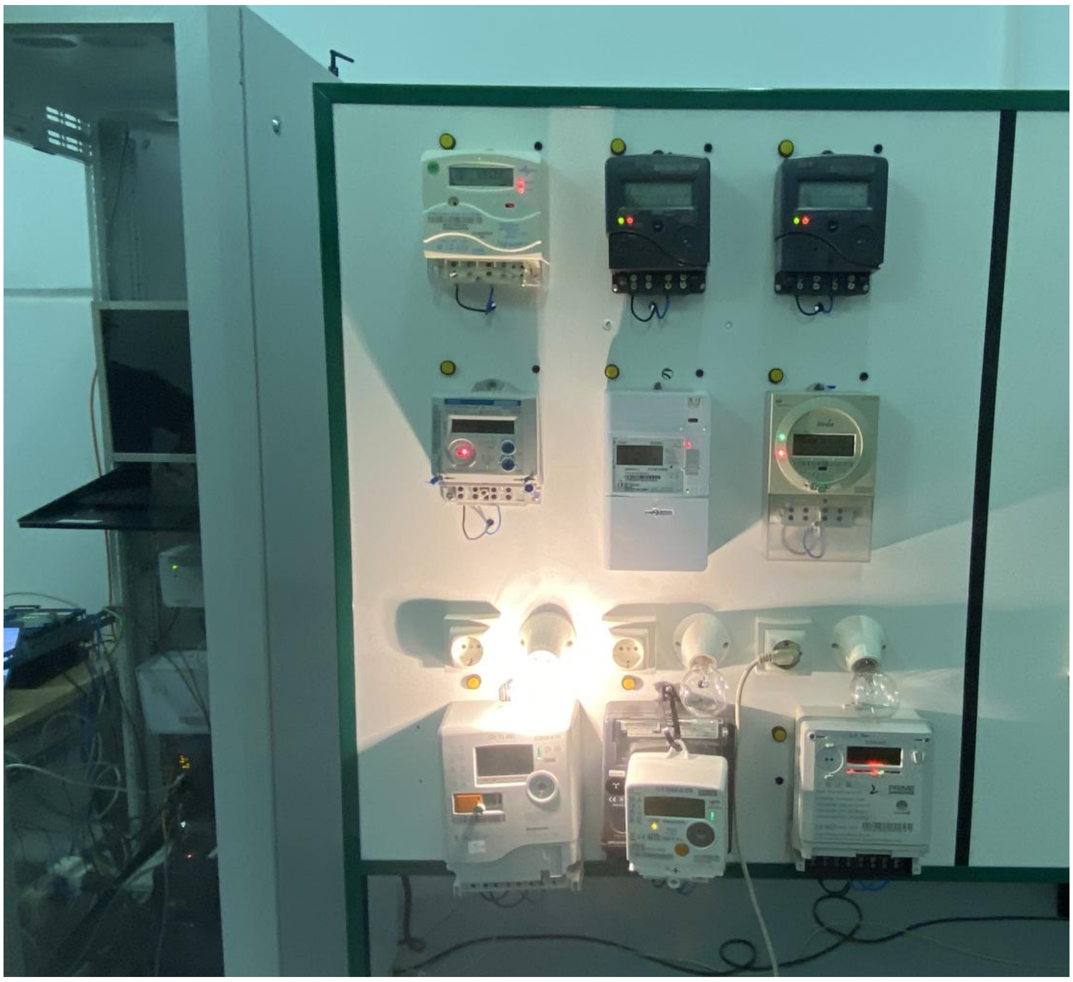
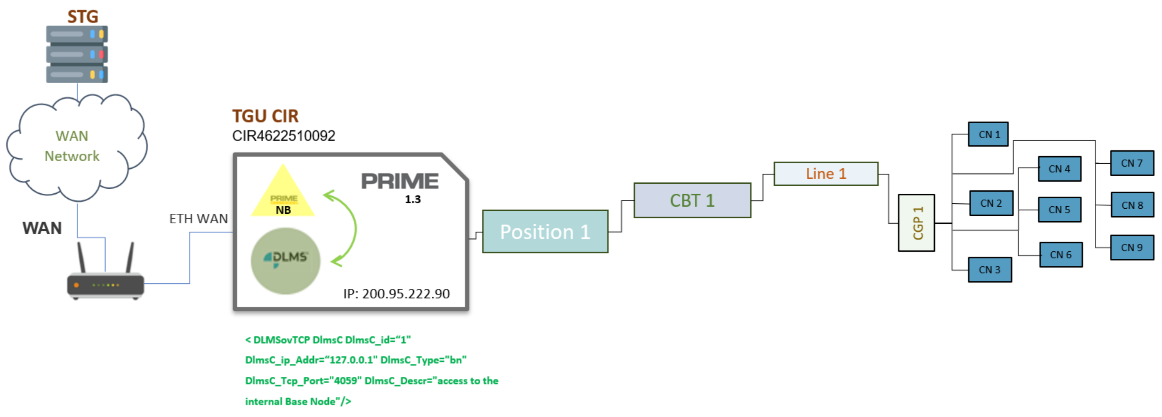


Figure 10: Picture of the laboratory setup for testing

On the other hand, it is possible to change the logical connections to repeat the test with the reference Circutor's Data concentrator that could be used for debugging in case that the origin of the problem is unknown. This configuration allows to know that the smart meters are correctly connected and respond to an already verified DC. This reconfiguration is shown in Figure 11, where the DC's internal DLMS over TCP is re-enabled and connected with its integrated base node that is already connected with the same smart meters.



```
< DLMSovTCP DlmsC DlmsC_id="1"
DlmsC_ip_Addr="127.0.0.1" DlmsC_Type="bn"
DlmsC_Tcp_Port="4059" DlmsC_Descr="access to the
internal Base Node"/>
```

Figure 11: Schematic of the data concentrator testing architecture in i-DE's laboratory

5.4.1.2 Field pilot

In the field pilot environment, two scenarios were evaluated to test the VDC under real distribution network conditions. The first scenario (Scenario 1) consisted of connecting the VDC to a single secondary substation in northern Spain equipped with an intelligent transformer to implement an On-Load Tap Changer (OLTC) regulation use case. While the second scenario (Scenario 2) involved connecting the VDC to multiple secondary substations, located at the center of Madrid, simultaneously to perform a scalability stress test, determining how many smart meters the VDC can manage in parallel. The secondary substations for this scenario were specifically selected between the ones with more supply points of Iberdrola i-DE (~ 1,000 smart meters).

The architecture for Scenario 1, showing the connection between the VDC (located at the laboratory of Iberdrola i-DE in Madrid) and the field secondary substation with an i-Trafo in i-DE's LV distribution network is shown in Figure 12.

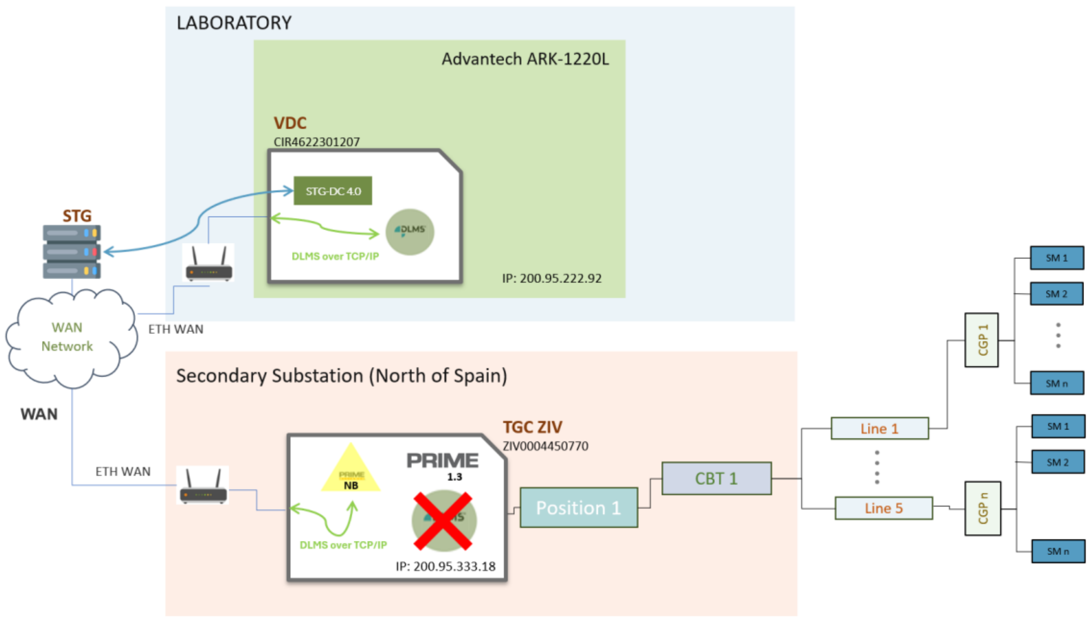


Figure 12: Schematic of the virtual data concentrator use case testing architecture in i-DE's LV distribution network.

On the other hand, the architecture for the scalability test of the VDC, consisting of connecting it to multiple substations is depicted in Figure 13. In the scalability scenario, the VDC application (running at the edge) theoretically supports up to 16 base nodes concurrently and up to 8 parallel communications. For this pilot, 12 big secondary substations were connected at once to push the limits of the system. These two field scenarios provided a practical context to assess VDC performance for both a specific automation use case and under high communication load.

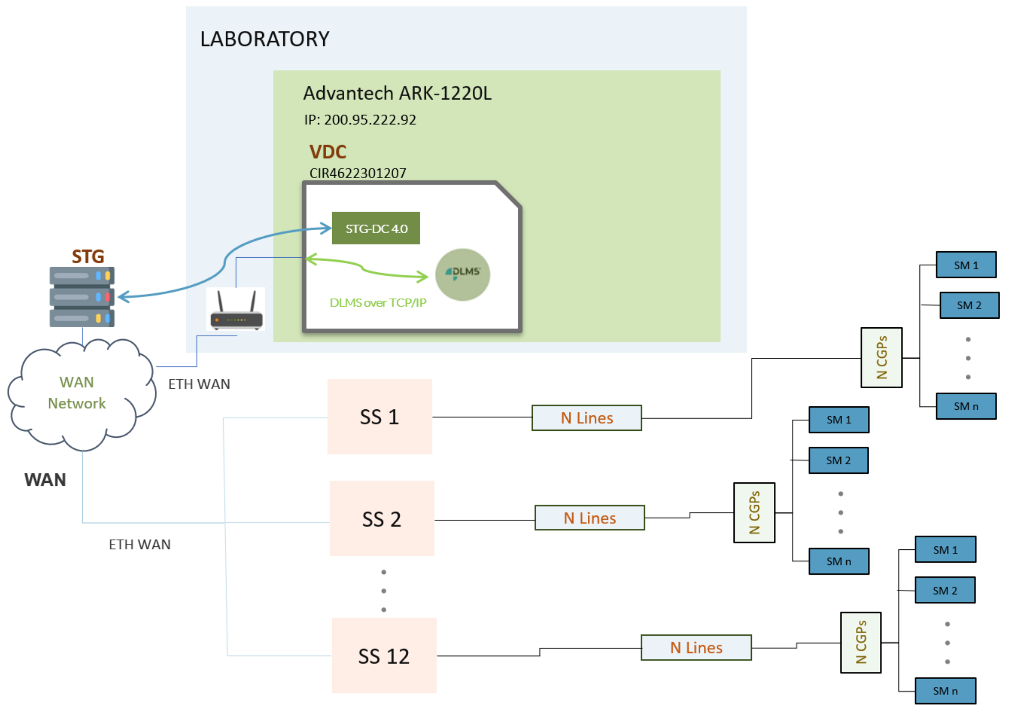


Figure 13: Schematic of the virtual data concentrator scalability testing architecture in i-DE's LV distribution network

5.4.2 TESTS

The tests that will be carried out to evaluate whether the virtual data concentrator is capable of performing the basic functions required of a fully integrated concentrator (hardware + software) can be divided into three distinct procedures, which must be executed in the chronological order presented below.

First, a configuration and connectivity test will be conducted. The second test will verify that the concentrator is able to receive basic measurements from the smart meters connected to it via PRIME. And third, it must be tested that the VDC can autonomously and periodically execute scheduled tasks, replicating the behaviour of physical concentrator.

A detailed explanation of the follow procedure to conduct these three tests is presented in the following subsections.

5.4.2.1 Configuration and connectivity tests – Laboratory Environment

The configuration and connectivity tests carried out in the laboratory environment start by requesting an S12 report to the VDC. This report contains the concentrator static and dynamic information, and its parametrization. This report allows the head-end, to check that the factory configuration of the device is as specified by i-DE and that correctly responds to the specified local IP address for local configuration equipment.

After the factory configuration is checked it is necessary to set up the concentrator with the correct parameters that ensure its correct operation. These parameters will allow the VDC to communicate with the RMS, to transfer the asynchronous files to a specified FTP server and to stablish DLMS communication over TCP with the PRIME base node among others. In order to set up all of this a B07 order needs to be sent to the VDC.

Prior to the B07 order for the VDC it is necessary to send a B07 order to the physical Circutor concentrator, that will serve as a PRIME base node. The purpose of this order is to disable the internal DLMS over TCP connection of this concentrator to allow its use as an external PRIME BN for the VDC tests. This disconnection was done by overwriting the internal DLMS TCP connection by selecting an empty IP address and changing the default port from 4059 to an arbitrary value. The specific XML order is presented below:

```xml
<Order IdReq="B07" IdPet="0" Version="4.0">
    <Cnc Id="CIR4622510092">
        <B07>
            <DLMSovTCP>
                <DlmsC
                    DlmsC_id="1"
                    DlmsC_ip_Addr=""
                    DlmsC_Type="bn"
                    DlmsC_Tcp_Port="1"
                    DlmsC_Descr="">
                </DlmsC>
            </DLMSovTCP>
        </B07>
    </Cnc>
</Order>
```

Next, a B07 order was sent to configure the three available RMS with which the VDC can be connected, including their IPs and their IDs, the FTP servers where the files will be sent and the DLMS over TCP/IP connection that should include the IP address of the physical data concentrator (200.95.222.90) and the port 4059. This particular order was sent using the web interface of the VDC and was captured using Wireshark via the laboratory's switch. The detailed configuration captured can be seen in Order B07.

Once the configuration is carried out a request of an S24 report is sent in order to ensure that the VDC is detecting the smart meters connected southbound. This report allows the VDC to retrieve the list of all the smart meters connected also proving that the connection DLMS over TCP was correctly done. Besides a complementary G24 report is requested to obtain a more detailed view of the identified smart meters in the S24 report.

In summary, these initial configuration and connectivity tests ensure that the VDC is ready to be tested in a deeper way and that it is able to stablish northbound and southbound communication.

5.4.2.2 Functional measurement tests – Laboratory Environment

This section evaluates the capability of the VDC to collect measurement data such as voltage, current or active and reactive power from the smart meters located in the laboratory. In order to test this functionality two simple scenarios will be considered, one where the measurements of an smart meter without any connected load are requested and the other one where an smart meter with a load connected is asked for its measurements. These requests are carried out asking for an S21 report to an specific smart meter.

In the first scenario the AUX1023030017, an smart meter from the Chinese provider Sanxing was requested to send an S21 report to the i-DE's RMS with no connected load. And in the second scenario an S21 report was requested to the SAG018630128, an smart meter from the French provider Sagemcom, which had connected an incandescent lamp of 60 W.

The results obtained are shown in 6.1.2.

5.4.2.3 Scheduled task tests – Laboratory Environment

Once commissioned, the VDC must be capable of executing a specific designed list of tasks i-DE uses for billing, supervision and asset maintenance. The data concentrators usually autonomously execute the following tasks:

• Task 1: This task requests a S05 report, that returns the daily billing values profiles. Every day at 00:10 it takes a daily register with absolute values per contract and tariff period of the previous day.

• Task 2: This task requests a S02 report, that returns the hourly incremental load profile. It takes the incremental defined in "Per" meter parameter values from the sixth magnitudes (2 active and 4 reactive) from a timing period usually equal or higher to one day of one or more meters. Typically for Iberdrola the "Per" parameter is 1 hour, therefore S02 will take the incremental hourly. This task is usually programmed at 00:10 and repeats every day.

• Task 3: This task requests a S04 report, that returns the Monthly billing values profile. It takes a monthly register (programmed billing) or when there is an end of billing (not programmed) with absolute and incremental values per contract and tariff period and it includes also the maximum value (incremental) of the import active power per tariff period and per contract with the timestamp. This task runs the first day of every month at 01:10.

• Task 4: This task requests a S09 report, that returns the registered meter events. The different types of events are defined on the data concentrator specification. This task is scheduled to be run at 03:00 every day. And helps to notice tampering, outages or changes in the configuration.

• Task 5: This task orders a Reboot of the DC. When the DC reboot by a T01 order execution, all running reports (due to scheduled tasks or STG report requests) are stopped. This task is automatically ran every fourteen days at 15:00, usually on Sunday to avoid major interruptions. This task is aims to prevent concentrator failures.

• Task 6: This task forces meter synchronisation ignoring parameters TimeDev and TimeDevOver. This task is usually scheduled daily at 22:00 to avoid major drifts between clocks.

• Task 7: This task requests a S17 report, that returns the registered concentrator events. It takes the event register of the concentrator depending on the time interval requested by the STG. The S17 report is collected daily at 22:50 and provides logs of internal alerts or communication errors.

• Task 8: This task requests a S24 and a S11 report. The S24 returns a table of the managed meters. It contains the most important information for each of the meters that have been detected by the Base Node and are managed by the DC. This information will indicate the Management System not only the existence of a specific meter in the network, but also the main data about its status and its availability. On the other hand the S11 report gets instant information from the PLC base node in the concentrator and registers PLC event from a determined date. The problem is that for the moment the S11 only returns information if the base node is integrated within the concentrator, so in the VDC case it is not expected to function. This task executes every day at 22:40.

• Task 9: This task requests a S14, that returns the voltage and current profiles of the supervisor meters. This task requires that the substation includes a LVS, which is not the case, so it is not expected to send the profiles. This task executes every day at 19:00.

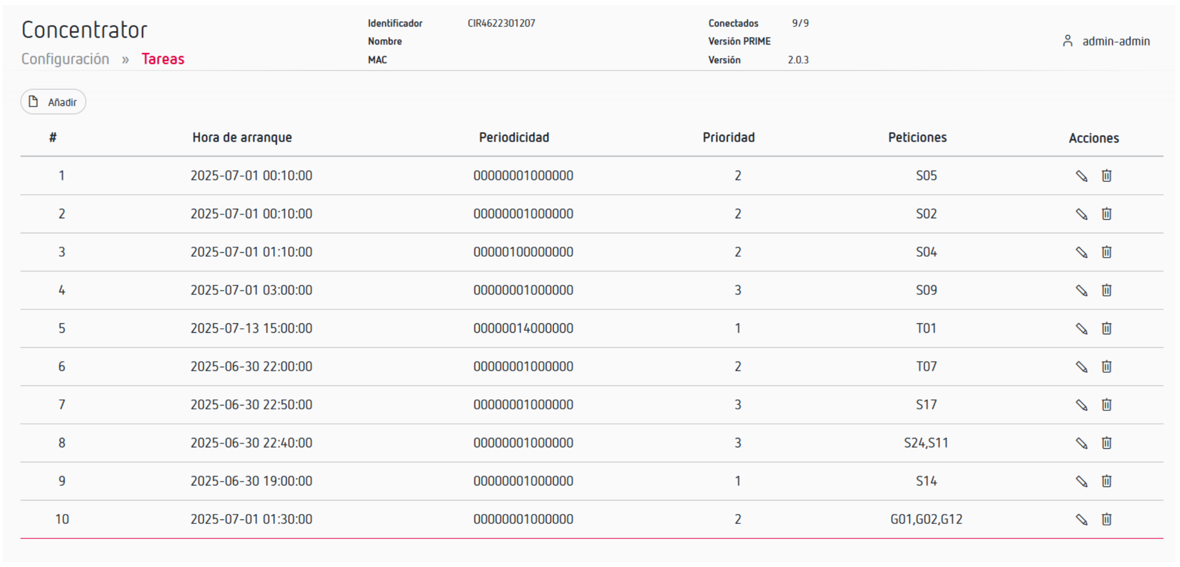
• Task 10: This task requests G01, G02 and G12 reports. G01 provides hourly communication statistics with meters. G02 provides daily communication statistics with meters. And G12 provides the historical record of the performance of the concentrator. This task is scheduled every day at 01:30 and helps to check the quality of communications.

In order to give a clear view of the scheduled tasks including the reports requested, the periodicity and the priority (0 beeing the highest and 4 the lowest) of each task a summary is provided in Table 4.

Task	***Request Type***	***Scheduled Time***	***Periodicity***	***Priority***
1	S05	00:10	Daily	2
2	S02	00:10	Daily	2
3	S04	01:10	Monthly	2
4	S09	03:00	Daily	3
5	T01	15:00	Every 14 days	1
6	T07	22:00	Daily	2
7	S17	22:50	Daily	3
8	S24 & S11	22:40	Daily	3
9	S14	19:00	Daily (supervisors only)	1
10	G01, G02, G12	01:30	Daily	2

Table 4: Periodic Tasks Scheduled in the VDC

These tasks can be scheduled by hand in the web interface provided by the VDC shown in Figure 14, or could be ordered from the RMS sending a B16 order.


Concentrator
Configuración » Tareas

Identificador CIR4622301207
Nombre
MAC

Conectados 9/9
Versión PRIME
Versión 2.0.3

admin-admin

Añadir

#	Hora de arranque	Periodicidad	Prioridad	Peticiones	Acciones
1	2025-07-01 00:10:00	00000001000000	2	S05	✎ 🗑
2	2025-07-01 00:10:00	00000001000000	2	S02	✎ 🗑
3	2025-07-01 01:10:00	00000100000000	2	S04	✎ 🗑
4	2025-07-01 03:00:00	00000001000000	3	S09	✎ 🗑
5	2025-07-13 15:00:00	00000014000000	1	T01	✎ 🗑
6	2025-06-30 22:00:00	00000001000000	2	T07	✎ 🗑
7	2025-06-30 22:50:00	00000001000000	3	S17	✎ 🗑
8	2025-06-30 22:40:00	00000001000000	3	S24,S11	✎ 🗑
9	2025-06-30 19:00:00	00000001000000	1	S14	✎ 🗑
10	2025-07-01 01:30:00	00000001000000	2	G01,G02,G12	✎ 🗑

Figure 14: Scheduled tasks in the web interface of Circutor.

5.5 USE CASE: OLTC REGULATION AND MONITORING

This section demonstrates that an edge-node implementation of a data concentrator (via containerization) can perform real-time control tasks in the field without needing to stream large volumes of data to the cloud, therefore avoiding latency issues that hinder immediate action. To achieve this, a virtualized setup was developed to continuously read smart meter measurements and to generate transformer tap change decisions based on feeder voltage levels using a simple regulation algorithm.

The OLTC regulation pilot was conducted on a secondary substation featuring an intelligent power transformer (i-Trafo). The substation's existing ZIV data concentrator device was used as an edge computing base node by disabling its native concentrator functionality and remotely connecting the VDC to it. This SS was chosen due to its reliable communication metrics (high successful read rate) with the downstream smart meters, ensuring a robust proof of concept for real-time control.

Since the selected secondary substation serves 175 smart meters in its LV network, a preselection of a representative set of smart meters was carried out in order to manage data effectively. The selection was based on the following criteria:

1. Only three-phase meters operating at 400/230 V were considered, as these are typically associated with higher-load installations or serve as supervisory meters that usually have single-phase meters connected downstream from them.
2. At least one meter from each of the five LV feeder lines outgoing from the substation was included, ensuring spatial coverage of the entire network downstream of the transformer, and capturing voltage conditions on all feeders.
3. All the meters known to be unreliable, that is the ones with permanent failures or frequent transient failures, were excluded to ensure the reliability of the data used for control decisions.

4. Finally, preference was given to smart meters supplying a large number of end users, so that their measurements reflect aggregated load impacts and critical network points.

As a result of this filtering using i-DE's confidential data base, the following five smart meters from were selected as most representative: ZIV0047922435, SAG0166016473, SAG0166016496, ZIV0041674575, and ZIV0047545478. These meters collectively provide a comprehensive view of the LV network's operating conditions and are instrumental for both the OLTC control algorithm and visualization in the Grafana monitoring platform.

Once the smart meters used as input data were selected, a custom Python-based application was developed on the edge node to implement the tap regulation logic. This application periodically issues S21 load profile requests (simulating the RMS polling S21 reports) to the VDC for the five selected meters. When the meter measurements are received (voltages, currents, power…), the program computes the average three-phase voltage across all monitored points and compares it to the nominal expected voltage (230 V) with a tolerance band (±3%). Based on this, a decision is made by a simple algorithm to adjust the transformer's tap, that returns a 1 if the average voltage is below the acceptable range (to raise the tap and increase voltage), a –1 if the average voltage is above the range (to lower the tap and reduce voltage), or a 0 if the voltage is within the acceptable band (no tap change required). The resulting tap change command is then logged along with the measured voltages. This algorithm runs every 5 seconds and sends all the collected data, including per-meter voltages and the OLTC decision at each interval, to an Influx time-series database on the edge. To have a more detailed view of how the report request, the algorithm and the commands to writ in the data base were carried out, the developed script is presented in Appendix III: Voltage Regulation Algorithm .

Finally, a containerized Grafana dashboard periodically queries this database to visualize the real-time voltage readings and tap position decisions. To provide a schematic overview of the logical connections of this setup and illustrate how the edge applications (virtual concentrator, control algorithm, database, and Grafana visualization) interface with the secondary substation and smart meters, a basic schema showing al the connections is presented in Figure 15. This edge-based architecture enables real-time monitoring and control (tap changer regulation in this case) directly at the substation level. The performance and outcomes of this OLTC regulation use case are discussed in section 6.2.

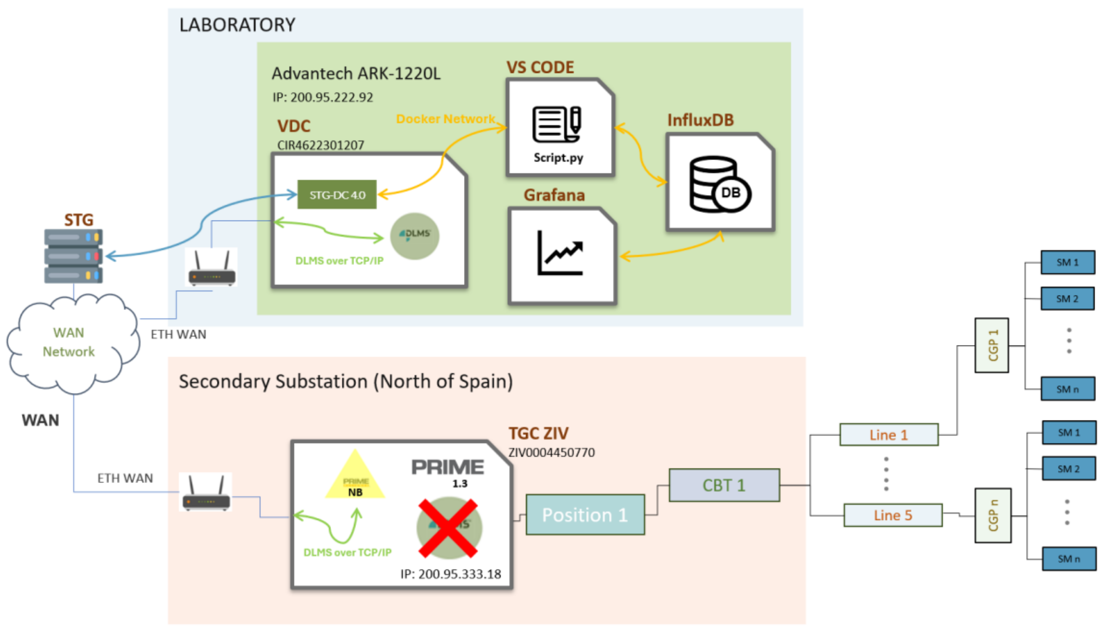


Figure 15: Schematic of the logic connections between the different containers in the Secondary Substation

Chapter 6. RESULTS ANALYSIS

6.1 VALIDATION TESTS RESULTS

6.1.1 CONFIGURATION AND CONNECTIVITY TESTS RESULTS

This section presents and analyzes the results obtained from the configuration and connectivity tests explained in section 5.4.2.1.

Regarding the configuration test, the request for the S12 report was successfully processed at the edge node level; that is, the VDC container log confirmed that the report had been generated correctly. However, the STG system rejected the report due to the absence of the ipCom2 and ipMask2 fields, which are mandatory according to the STG specification. However, these fields are irrelevant in the context of a virtual concentrator, and their omission does not imply a failure in the design, since the specification was developed thinking about physical concentrators, which typically include two communication ports and therefore require these parameters.

Despite this discrepancy, the VDC remains operational and functionally correct and was able to generate a coherent S12 report, extracted from the error folder of the FTP server. As shown in Report S12 (Pre configuration) the parameters ipCom2 and ipMask2 are missing as mentioned above, but the configuration related to the STG and FTP services appears to be correctly defined by default, including IP addresses, ports, and user credentials. Moreover, the DLMS over TCP section are unconfigured, as expected, since the VDC does not include an internal node base.

After checking its default configuration, and sending the B07 order to the VDC, as explained in 5.4.2.1, to configure the DLMS over TCP communication, another S12 is sent back to let know the user that the configuration was correctly carried out. Report S12 (Post configuration), shows the parameters and configuration of the DLMS over TCP where can be seen that the DLMS was enabled, and that the IP configured was the one of the physical Circutor data concentrator (200.95.222.90) and the port 4059.

Moreover, both reports S24 and G24 where successfully requested from the RMS and received to the FTP server, and correctly returned a list of the nine smart meters that were used as testbench in the laboratory as shown in Table 5. Report S24 and Report G24 show the list of the registered meters in the VDC alongside with their communication status, and detailed parameters from each meter such as MAC address, firmware version…etc

Smart Meter	*ID*
1	AUX1023030017
2	ELS0032114924
3	ITE0141020666
4	ORB0000803879
5	SAG0125322720
6	SOG0000502076
7	SAG0186301287
8	ZIV0039808892
9	ZIV0044603022

Table 5: List of the smart meters' IDs connected in the Laboratory environment.

6.1.2 FUNCTIONAL MEASUREMENT TESTS RESULTS

This section presents the results obtained following the procedures explained in 5.4.2.2.

As expected, both S21 requested reports were successfully sent. For the first scenario, the report returned by the VDC shows zero values for current and power across all phases and shows 226 V in phase 1 since the smart meter is single phase connected. The obtained report reflecting the unloaded meter is provided below:

```
<Report xmlns="http://stgdc/ws/S21" IdRpt="S21" IdPet="0" Version="4.0">
    <Cnc Id="CIR4622301207">
        <Cnt Id="AUX1023030017">
            <S21 Fh="20250630153235000S" Ca="0" I3="0.0" L1v="226" L1i="0.0"
Pimp1="0" Pexp1="0" Qimp1="0" Qexp1="0" PF1="1.000" Ca1="0" L2v="0" L2i="0.0"
Pimp2="0" Pexp2="0" Qimp2="0" Qexp2="0" PF2="1.000" Ca2="0" L3v="0" L3i="0.0"
Pimp3="0" Pexp3="0" Qimp3="0" Qexp3="0" PF3="1.000" Ca3="0" PP="1,0,0" Fc="5"
Eacti="0" Eanti="0" AIa="1" AEa="0" R1a="0" R2a="0" R3a="0" R4a="0"></S21>
        </Cnt>
    </Cnc>
</Report>
```

On the other hand, the S21 report for the scenario 2, shows an imported active power of 60 W through phase 1, a voltage of 227 V and an instantaneous current of 0.2 A. As the connected load is pure resistive, the power factor is 1 and the reactive power exported and imported is zero. The report in XML format is presented below as a prove of the correct functioning of the device.

```
<Report xmlns="http://stgdc/ws/S21" IdRpt="S21" IdPet="0" Version="4.0">
    <Cnc Id="CIR4622301207">
        <Cnt Id="SAG0186301287">
            <S21 Fh="20250705183004000S" Ca="1" I3="0.2" L1v="227" L1i="0.2"
Pimp1="60" Pexp1="0" Qimp1="0" Qexp1="0" PF1="1.000" Ca1="1" L2v="0" L2i="0.0"
Pimp2="0" Pexp2="0" Qimp2="0" Qexp2="0" PF2="1.000" Ca2="0" L3v="0" L3i="0.0"
Pimp3="0" Pexp3="0" Qimp3="0" Qexp3="0" PF3="1.000" Ca3="0" PP="1,0,0" Fc="5"
Eacti="1" Eanti="0" AIa="150765" AEa="10312" R1a="10353" R2a="1247" R3a="81"
R4a="871"></S21>
        </Cnt>
    </Cnc>
</Report>
```

6.1.3 SCHEDULED TASKS VALIDATION RESULTS

This section presents and analyses the results obtained following the procedures explained in 5.4.2.3.

Each one of the explained tasks was individually analysed to ensure the correct operation of the VDC as follows:

Task 1 (S05), that was scheduled to be launched at 00:10 was successful generated by the VDC as can be seen in the logs generated by the VDC app on the edge node shown in Logs S05, however the development RMS did not correctly store the report due to issues in the RMS development version.

Task 2 (S02) was correctly sent at the scheduled time and was correctly uploaded to the FTP server. The XML report showing the hourly load profile can be found in Report S02, where the incremental active import of all the smart meters is shown as zero except for the smart meter number 7 that has the load connected.

Task 3 was correctly sent at the scheduled time (01:10) by the VDC as can be seen in the logs of the edge node Logs S04. However, as the case of Task 1, the FTP server failed to save the report and couldn't be obtained.

Task 4 (S09) was correctly sent at the scheduled time and was correctly as shown in Logs S09. This log shows that the task was completed successfully for all the smart meters except for the ZIV0044603022, that gives errors with all the scheduled tasks. When a scheduled task fails to send the report for one meter due to temporary failure, it retries every 30 minutes as shown in all the logs of Appendix II: Scheduled Tasks Tests.

Moreover, the scheduled reboot of the concentrator (Task 5) and the meter synchronisation were correctly done at 15:00 on Sunday and at 22:00 respectively. The logs of the VDC confirming its correct functioning are shown in Logs T01 and Logs T07.

Task 7, which provided internal concentrator logs could send the report but was not saved by the server. So relevant logs proving its functioning are provided in Logs S17.

Task 8, that included a S24 request an a S11 request was completed successfully by the VDC. In this case both reports are available and it is possible to see the smart meters list in

Report S24 and that the Report S11 shows a report with empty information about the base node since there is no integrated base node in the edge node.

As expected, Task 9 thrown an error as shown in Logs S14, since there is no available LVS at the laboratory. This resulted in a correct rejection from the VDC when attempts to request this task was done from the RMS.

Finally, task 10 successfully requested the G01, G02 and G12 reports, which were received an stored appropriately. It is possible to extract the following from these reports:

The Report G01 presents hourly communication availability metrics. Each report entry corresponds to a specific hour (Fh), and includes:

- Amed, Amax, Tot: All equal to 9, indicating that 9 communication attempts were made, and all of them were successful.
- Aperc: 100.00%. That confirms full availability.

This means that the VDC maintained 100% communication availability during the reporting period, which is expected in a laboratory environment.

The Report G02 details the communication status of individual meters (Cnt) connected to the concentrator. Each report entry includes:

- Atime: 1440 minutes (24 hours), indicating full-day connectivity.
- Nchanges: 0, meaning no communication state changes occurred.
- Aconc: 1440, confirming continuous connection.
- Atimeperc: 100.00%, again indicating full availability.

Finally, the Report G12 provides hourly system resource usage data for the concentrator. Each report entry includes:

- Cpu, Ram, Flash, PerStor: Indicate consistent resource usage (e.g., CPU = 2, RAM = 7).

- EthRx, EthTx, SerRx, SerTx: All zero, suggesting no Ethernet or serial traffic during the period.
- Temp: Constant at 23°C, indicating thermal stability.

6.1.4 SUMMARY OF RESULTS

Based on the tests results presented above, it can be concluded that the VDC solution is perfectly feasible from the technical point of view, being able to execute all the defined tasks traditionally performed by legacy data concentrators. A structured summary of the achieved results is provided in Table 6, which consolidates the outcome of each test and highlights the compliance of the VDC with expected functionalities.

The minor issues observed during testing, such as the unsuccessful processing or storage of some reports like the S05 or the S17, were primarily due to the limitations inherent to the STG development environment used for testing rather than the VDC itself. This means that if the fully operational field STG environment was used these issues would likely not arise, because it is frequently updated and maintained.

Furthermore, errors such as the rejection of the S12 report while configuring the VDC due to missing mandatory fields (specifically ipCom2 and ipMask2) are easily addressable. Such errors could be solved either by slightly adjusting the existing DC specification or by asking the developers of the VDC application to populate these required fields with dummy values of mask and IP address that would not serve any operational purpose but would ensure compliance with the existing validation schemas already installed in the FTP server.

Overall, the conducted laboratory tests clearly validate the technical viability of deploying VDCs as substitution of physical DCs, making minor adaptations to existing specifications as explained.

Test Category	***Test***	***Executed Successfully***	***Received/Stored in STG or FTP***
Configuration & Connectivity	S12 – Factory Configuration Report	✓	✗ *(Missing fields: ipCom2, ipMask2)*
	B07 – Disable Internal DLMS	✓	–
	B07 – Initial Configuration	✓	–
Functional Measurements	S24/G24 – Meter Discovery	✓	✓
	S21 – Meter 1 (No Load)	✓	✓
	S21 – Meter 7 (Load: 60 W)	✓	✓
Scheduled Task Validation	S05 – Daily Billing Report	✓	✗ *(STG development issue)*
	S02 – Hourly Load Profile	✓	✓
	S04 – Monthly Billing Closure	✓	✓ *(STG development issue)*
	S09 – Meter Event Log	✓	✓
	T01 – Reboot Order	✓	✓
	T07 – Forced Time Sync	✓	✓
	S17 – Concentrator Event Log	✓	✓
	S24/S11 – SM List and BN Status	✓	✗ (S11 only: No BN)
	S14 – Voltage and Current Profile	✓	✗ (no LVS)

	G01 – Communication Statistics (Hourly)	✓	✓
	G02 – Communication Statistics (Daily)	✓	✓
	G12 – DC Performance History	✓	✓

Table 6: Summary of VDC validation tests results

6.2 USE CASE RESULTS

This section presents the results obtained from the OLTC regulation and monitoring use case executed on the edge node. The containerized Grafana application was used to monitor both the control algorithm's decisions and the electrical measurements in real time. In particular, the system tracked the tap change commands determined by the OLTC algorithm alongside the three-phase voltages reported by the five representative smart meters, as well as additional power metrics per phase.

As shown in Figure 16, the OLTC control algorithm generates tap adjustment commands in response to the observed voltage levels. Initially, the transformer tap position was at nominal voltage. The algorithm correctly identified deviations beyond the tolerance band and issued commands to lower the tap to bring voltages within range. However, since the physical tap changer could not be actuated in the field pilot (due to operational constraints and lack of permission to send actual control signals to the intelligent transformer), the control loop was not closed. Consequently, the algorithm continued to recommend a tap decrease repeatedly, as the measured voltages remained above the nominal range without an actual tap change taking effect.

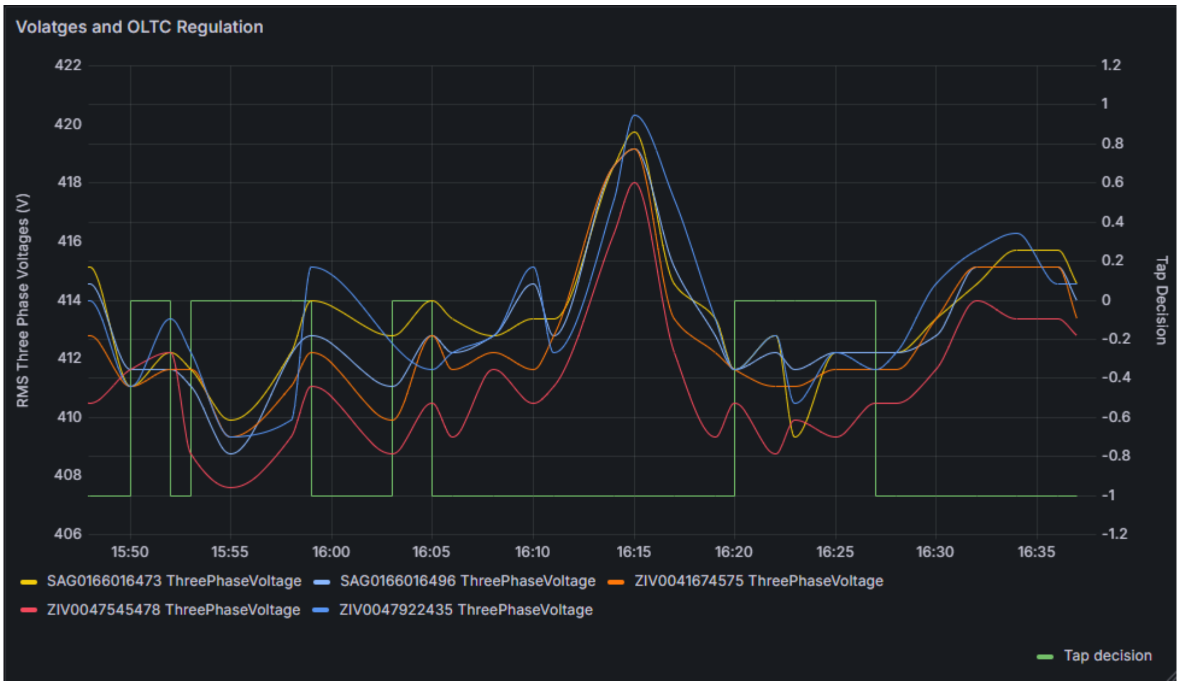


Figure 16: Plot of tap decision and real time three phase voltages from smart meters

Figure 17 and Figure 18 illustrate the capability of the edge platform to continuously monitor grid measurements from the selected meters. Figure 17 displays the real-time phase-to-neutral voltages (L1v, L2v, L3v) for each of the five monitored meters. This multi-meter voltage trend plot confirms that the system successfully acquires and streams high-resolution voltage data from diverse points in the network.

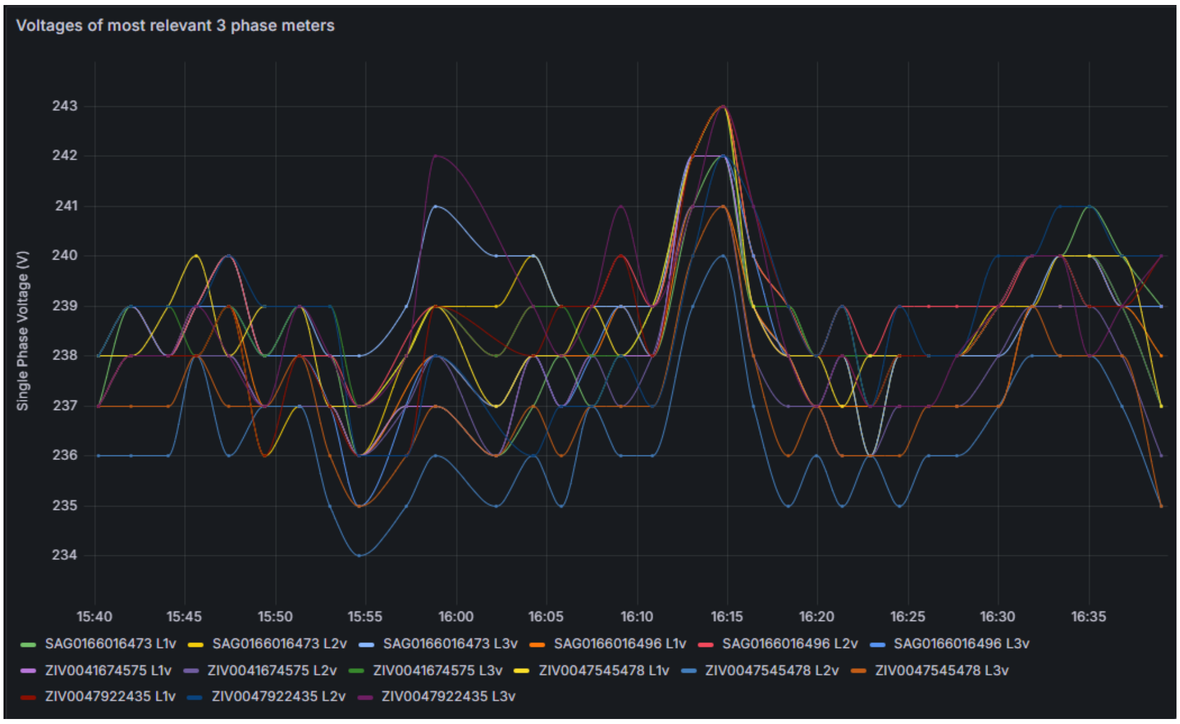


Figure 17: Single phase voltages of the most representative meters.

Figure 6 shows a summary of active and reactive power measurements (both imported and exported power) from the same set of meters over time. The stable visualization of these power metrics in real time further demonstrates the effectiveness of the edge computing approach for distribution network monitoring.

Overall, this use case highlights one of the many new functionalities enabled by virtualizing the data concentrator at the network edge. The edge-based VDC platform performed local voltage regulation decisions and high-frequency data acquisition without relying on cloud computing. This not only reduces the need for massive data transfers and cloud processing resources, but also makes possible real-time control actions (such as transformer tap adjustments) that would be impractical under cloud latency constraints.

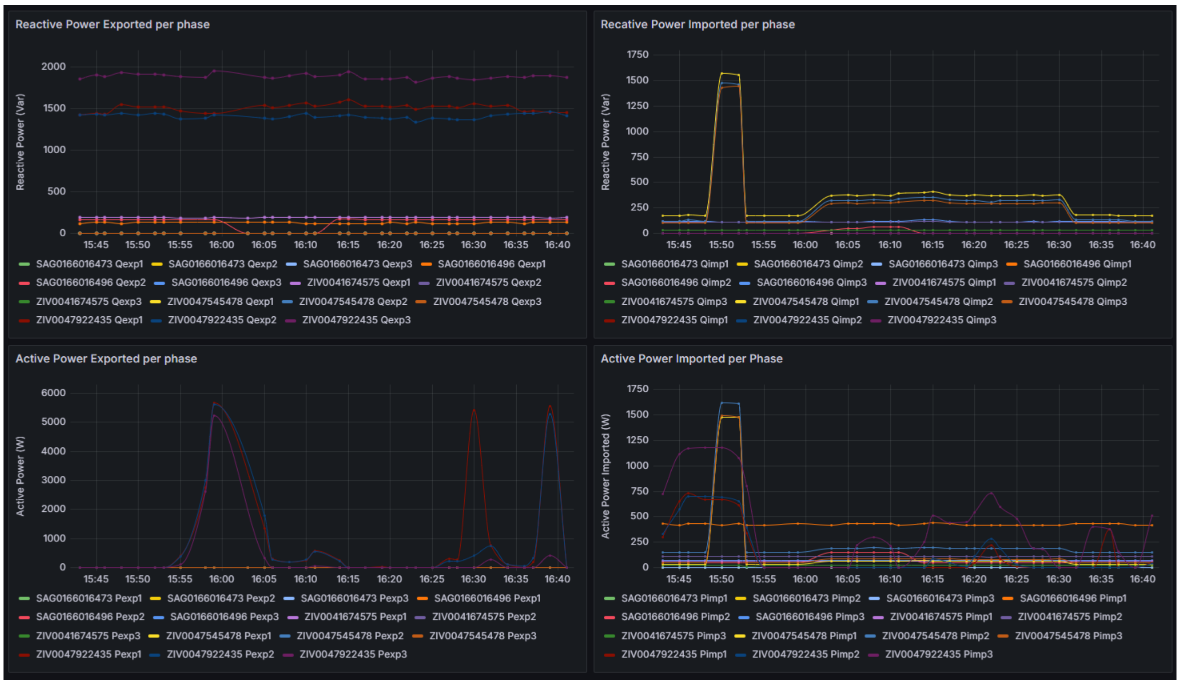


Figure 18: Active and Reactive Power measurements monitoring from smart meters

6.3 ECONOMIC IMPACT

6.3.1 CONTEXT AND DEPLOYMENT ALTERNATIVES

Iberdrola is currently carrying out a project called PRADA that aims to transition from PRIME 1.3.6 to PRIME 1.4 as well as upgrading the remaining legacy smart meters to the 2.0 version. This upcoming changes for the next regulatory period (01/01/2026 –31/12/2031) provide an opportunity to modernize the data concentrators in the CTs, since PRIME BNs need to be changed as well as legacy concentrators.

In this context, this section provides an economic analysis comparison between two different scenarios. The first scenario, (Base Case solution for now on) involves deploying new TGUC devices (traditional data concentrator with integrated PRIME BN and LVS) across the SSs. And the second scenario (Edge Case for now on), proposed in this project, involves deploying modern edge nodes running virtualized applications instead. Since, the only available virtualized solution for now is the VDC, this analysis aims to study whether this new solution can be more cost-effective that the conventional TGUC deployment.

6.3.2 COST ASSUMPTIONS FOR CAPEX AND OPEX

The economic analysis is based in cost estimates and realistic assumptions based on internal documentation. The estimated values of CAPEX and OPEX are provided in Table 7 and Table 8 respectively. Table 7 identifies the required hardware and software components and its unitary costs for both scenarios and assumes a 15-year amortization period for each device. Besides all the capita elements presented in the table are assumed to be remunerated investments under current regulation.

The CAPEX items include:

- TGUC: Traditional data concentrator, which includes a PRIME BN and a LVS. A unit will be need for each transformer position of the SS in the Base Case.
- Edge node: The industrial computer that contains the containerized applications. A unit will be required for each SS in the Edge Case.

- PRIME BN: External PRIME base node. A unit will be required per transformer position of the SS in the Edge case.
- Low Voltage Supervisor: External LVS. A unit will be required per transformer position of the SS in the Edge case.
- VDC License: Software license of the VDC application. A unit will be required per SS in the Edge Case.
- DPP License: Software license of the Edge Distributed Processing Platform. A unit will be required per edge node in the Edge Case.
- Equipment Installation: represents the one-time cost of deploying equipment in a SS. This cost is applied in both cases and is considered part of the investment.
- DPP Deployment represents the one-time cost of deploying the edge platform. It is assumed to be done the first year.

Both licenses will be treated as capitalize intangible assets with 15 years amortization as if it were included with the hardware.

Device	***Cost***	***Unit***
TGUC	450.00	€/Un
LVS	70.00	€/Un
Edge Node	300.00	€/Un
NB PRIME	100.00	€/Un
License VDC	250.00	€/Instance
License DPP	175.00	€/node
Equipment installation in SS	170.00	€/SS
DPP Deployment	445,000.00	€

Table 7: CAPEX

On the other hand, Table 8 summarizes the operational costs and estimated efficiencies that the new solution presents, including its value and unit. The OPEX components and efficiencies include:

- Intervention in SSs: represents the total cost of one intervention in a SS. This cost is applied in both cases.
- O&M interventions in SS: represents an estimate of the percentage of the total of SSs that will be intervened in a year. This cost is applied in both cases.
- Reduction in Edge interventions: represents an estimate of the reduction of interventions per year that the edge solution will entail.

Concept	***Value***	***Unit***
Intervention in MV/LV Substation	140.00	€/SS
O&M interventions in SS	3%	% of SS per year
Reduction in Edge interventions	30%	% of interventions per year

Table 8: OPEX

6.3.3 Deployment Volume and Rollout Modeling

In order to calculate the volume of devices to be deployed internal data regarding the number and type of SSs was consulted. The collected data is summarized in Table 9, showing that i-DE operates approximately 102,000 SSs with about 22 % of them containing two or more transformer positions resulting in approximately 124,368.00 transformer positions in total.

Classification	***SSs***	***SS with 2 or more positions***	***Positions MT/LV***
Total	102,054.00	22,314.00	124,368.00
Urban	55,819.00	22,314.00	78,133.00
Rural	26,185.00	0.00	26,185.00
Scattered rural areas	20,050.00	0.00	20,050.00

Table 9: Number and type of SSs in terms of location and transformer positions.

This distinction is important because both scenarios include devices that scale differently as shown in Table 10, where the number and type of devices that need to be deployed in each case is presented.

Deployed Devices	***TGUCs***	***Edge Node***	***PRIME BN***	***LVS***	***VDC License***	***DPP License***
Base Case	124,368	-	-	-	-	-
Edge Case	-	102,054	124,368	124,368	102,054	102,054

Table 10: Number of devices deployed on each scenario.

Furthermore, in order to model a realistic deployment of devices a Weibull distribution, with shape parameter α = 2 and scale parameter β = 2.7, will be used to represent a gradual deployment that reflects practical constraints that usually lead to a poor deployment the first year followed by a ramp up in the third or fourth years and finally slows down at the end of the period. The resulting annual deployment estimations as well as the accumulated deployment following this distribution are presented in Figure 19, showing the percentage of devices estimated to be deployed each year.

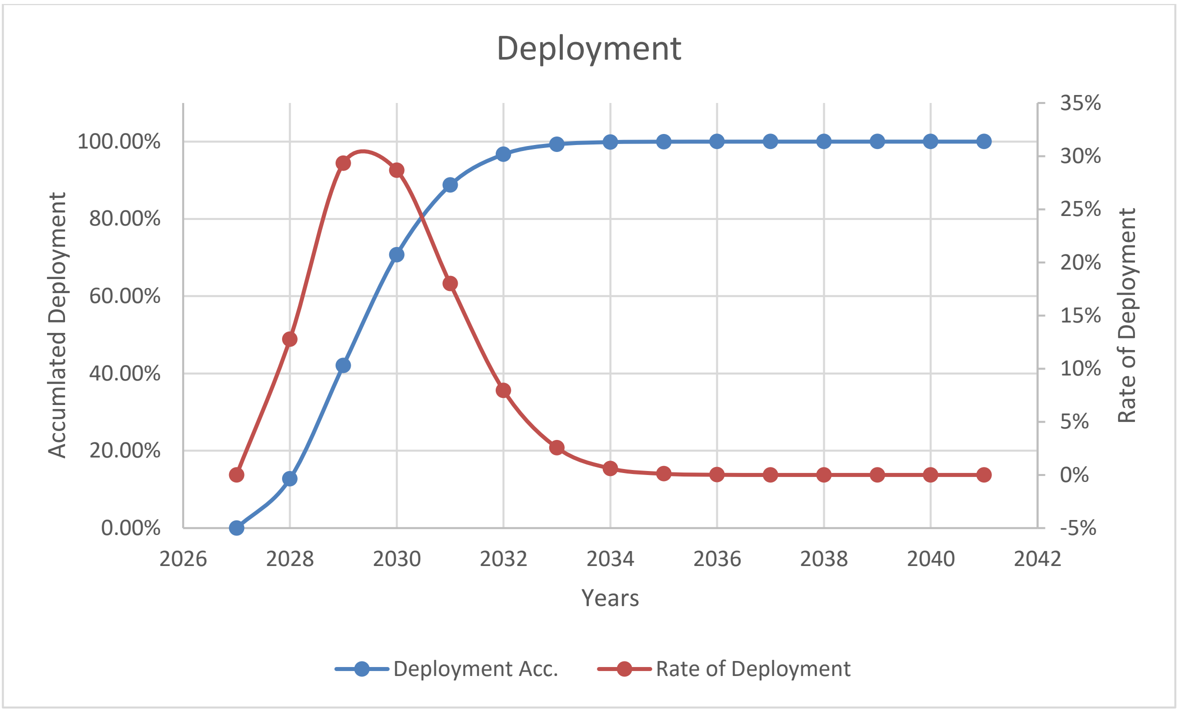


Figure 19: Estimated deployment of Data Concentrators by i-DE for the next regulatory period

6.3.4 Regulatory Remuneration Model

To properly evaluate the economic impact of this deployments it is necessary to understand the current rules of remuneration established by the Spanish regulator (CNMC). As explained in [25], new assets put into service by electric utilities after 31st December 2017 will earn a regulated remuneration for 15 years from two years after commissioning. Therefore, the cost-benefit analysis will assume the aforementioned regulatory framework and the most recent determination for the WACC that for the moment is 6.46 % as published by the CNMC. This value will be used as the discount rate and the remuneration rate of the investment.

Additionally, the formulas used to calculate the expected remuneration for each year are presented and briefly explained bellow and were extracted from [25].

$$R_n = \sum (A_n + RF_n) \qquad \text{Eq1}$$

$$A_n = \frac{VI}{VU} \qquad \text{Eq2}$$

$$RF_n = VN_n * TRF \qquad \text{Eq3}$$

$$VN_n = VI - (k-2) * \frac{VI}{VU} \qquad \text{Eq4}$$

Eq1 represents the total remuneration each year (n) and it is composed by the retributed amortization of the asset in the year n and the retribution because of the investment. Eq2 further defines the amortization as the value of the initial investment divided by the amortization period (15 years). Eq3 and Eq4 define the remuneration due to the investment and particularly Eq4 shows how the regulator remunerates the investment on the assets with two years lag from the commissioning date.

Since Spanish regulation establishes a fixed remuneration for OPEX, in this analysis they will be treated as maintenance costs directly affecting the cash flow and assuming that any reduction in the operation and maintenance costs translates as real savings for i-DE. Additionally, it will be assumed that the investment will be completely financed by i-DE with no reliance on externa loans.

6.3.5 COST-BENEFIT ANALYSIS RESULTS

Based on the assumptions from subsection 6.3.4, the cash flow for each year and deployment scenario was computed over the 15 years period, which allowed to compare both investments until the devices were fully amortized. These cash flows alongside with the incurred costs and the investments done for each scenario are presented in Figure 20 and Figure 21. Where the yearly breakdown of costs and benefits for the Base Case and the Edge Case are respectively outlined. In both cases, the outflows comprehend the capital investments made each year, that includes the installation of devices and the licenses, and the OPEX incurred, that includes maintenance interventions that grow with the number of units in service. On the other hand, the inflows comprehend the calculated regulatory remuneration. These charts allow to see how each project initially requires net spending and later yields positive cashflows through the years until the investment is recovered.

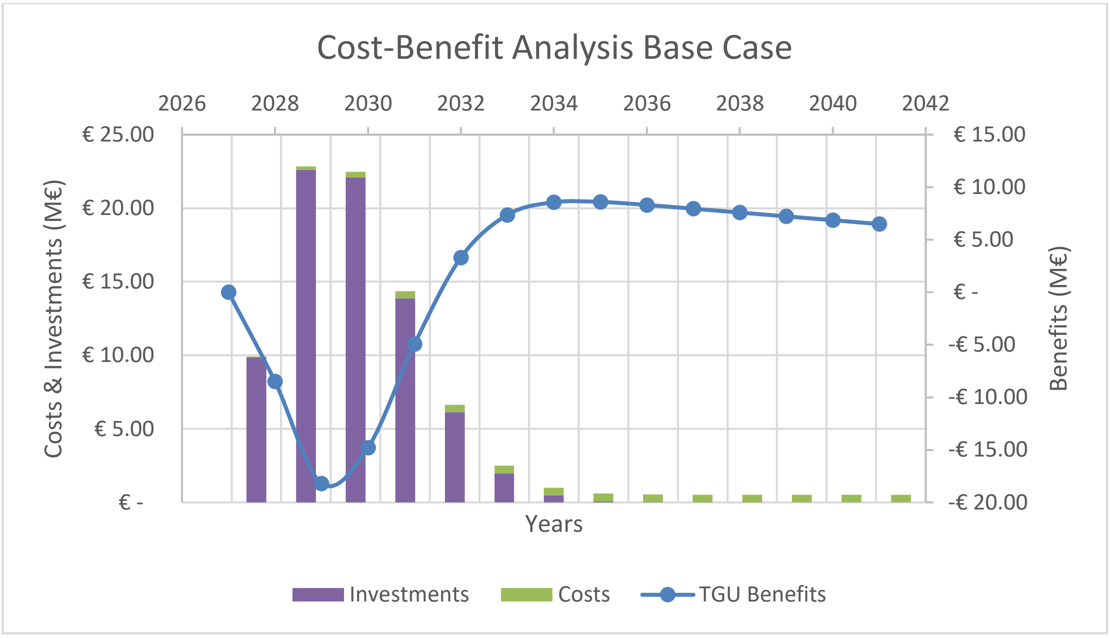


Figure 20: Cost-Benefit Analysis Base Case

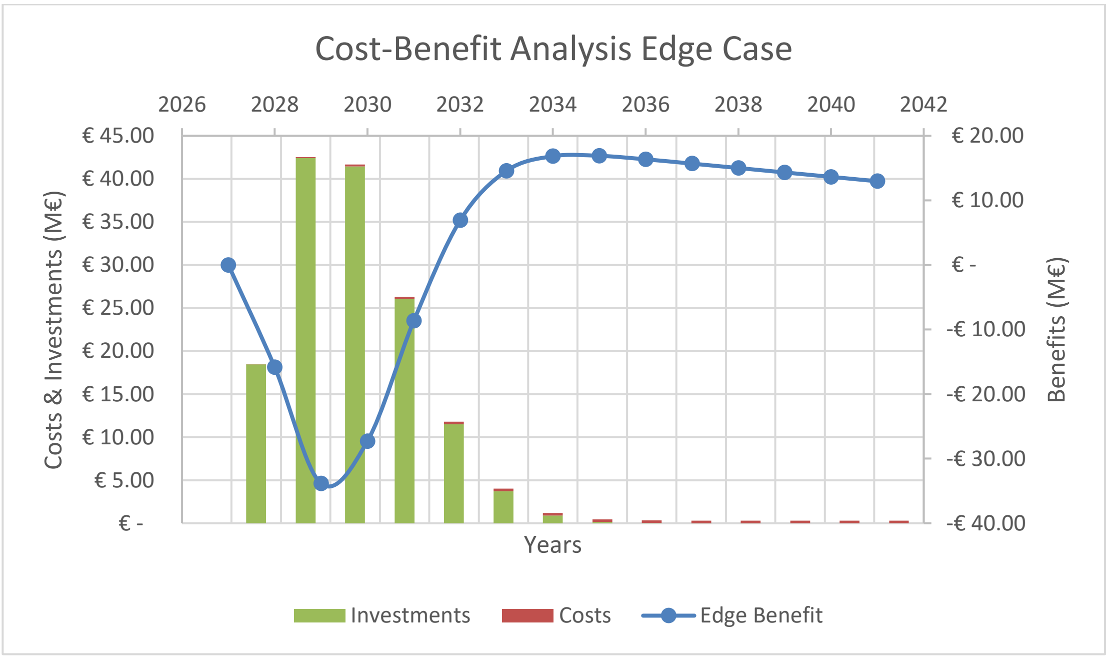


Figure 21:Cost-Benefit Analysis Edge Case

By the end of the period analysed both scenarios yield a positive NPV, nevertheless this depends on the deployment curve chosen, since the Weibull distribution used does not allow to see the complete cash flow in 15 years (it would be needed to analyse 17 years from last deployment to have a view of the full cash flow, that is 15 years of amortization plus the 2 years the CNMC delays on the remuneration). Since, the intent of this analysis is to compare both deployments there is no point to extend the analysis for more than 15 years.

The summary of the financial results is provided in Table 11, were key economical metrics are compared. In particular, the Net Present Value (NPV) and the Internal Rate of Return (IRR) are calculated, to determine which investment is more suitable for Iberdrola i-DE.

C&B Analysis	***NPV***	***IRR***	***Decision***
Base case	5,234.11 €	6.462 %	Profitable
Edge Case	5,185,432.29 €	7.584 %	Profitable

Table 11:Cost-Benefit Analysis including NPV and IRR for each scenario.

As shown in Table 11, both NPVs for 15 years cashflows are positive given a deployment curve as described in Figure 19, confirming that both projects are profitable in less than 15 years. Besides the IRR is greater than 6.46% which indicates that both projects would generate a return above the regulatory WACC.

However, when comparing both investments, it is possible to see that the Edge Case is more cost-effective than the base case, which could be counter intuitive given that the Edge case requires extra components. This is based on the assumption that both the application license and the distributed platform license would be fully remunerated by the regulator. Under this assumption, the current regulatory framework established by the CNMC, which favours capital expenditures, results in a significantly higher investment by Iberdrola, as the company would bear the cost of the licenses. Consequently, the expected return is also higher.

On the contrary, if we assume that the regulator does not recognize the licenses as part of the CAPEX, or remunerates their cost partially, the attractiveness of deploying edge nodes is substantially reduced due to the lower return on investment as shown in Table 12, where two alternative scenarios where calculated. The first scenario assumes that the regulator remunerates 50% of the licenses costs and the other one assumes that no remuneration is provided.

This analysis raises the question of what percentage of the licenses Iberdrola i-DE would need to request remuneration for, in order to justify choosing the edge computing solution over the current TGUC-based approach. To address this, a study was conducted using Excel's Goal Seek tool to determine the percentage of licenses that would need to be remunerated so that the Internal Rate of Return (IRR) of both scenarios would be equal. The result indicated that 86% of the license cost would need to be covered. This outcome is presented in Table 12.

C&B Analysis	***NPV***	***IRR***
Base case	5,234.11 €	6.462 %
Edge Case (50 % Remuneration)	-12,947,412.67 €	4 %
Edge Case (0 % Remuneration)	-31,080,257.63 €	-1 %
Edge Case (86 % Remuneration)	5,234.11 €	6.462 %

Table 12: Cost-Benefit Analysis including NPV and IRR if licenses are partially or not remunerated.

Finally, it is important to note that the CNMC intends to replace its remuneration model based on CAPEX + OPEX, for one where the OPEX is also taken into account called TOTEX. This change in the model could lead to drastic changes in these analysis results. Besides, it should be also considered that the Edge node case economic appeal could improve if more applications apart from the VDC were added such as a virtualized Remote Terminal Unit.

6.4 SCALABILITY VALIDATION

A brief scalability evaluation of the virtual data concentrator was carried out to determine its capacity in handling multiple simultaneous connections and a large population of meters. Specifically, an assessment was carried out to determine the maximum number of DLMS over TCP/IP sessions that the VDC is capable of maintaining in parallel, as well as the total number of smart meters with which it can sustain continuous communication.

The VDC software is theoretically designed to support up to 16 base node connections at once, with up to 8 concurrent communication threads. In the scalability test, twelve secondary substations in the Madrid city center (within i-DE's service area) were connected concurrently to a single VDC instance. Each of these substations has more than 800 meters, stressing the system with a substantial meter count. In total, the VDC discovered 7128 meters across the 12 substations. It successfully established live connections with around 6008 meters at peak, although some meters were connecting and disconnecting intermittently. The system maintained approximately 5900 meter connections stably on average during the test.

In addition to sustaining connections, the VDC's ability to perform data acquisition tasks at scale was examined. The primary challenge encountered was due to the meters' cybersecurity requirements. Most of the meters are "secure" devices that must exchange encryption keys with the RMS before responding to data requests. But, since the test environment used a development STG that did not hold the field meters' keys, the VDC began issuing iterative S31 security key requests for each meter. The RMS, lacking the necessary keys, continually requested them from the meters, which saturated the communication channel with repeated key exchange attempts. Despite this issue, it was possible to schedule certain readout tasks (such as profile read S05 and instantaneous read S02 requests), that were successfully sent by the VDC to all meters as shown in the logs presented below. As expected, only the few non-secure meters responded with the requested data, while the secure meters did not return reports (because of the unresolved key exchange). This outcome highlighted that the VDC's scalability in practice was constrained by the test environment configuration, rather than by the concentrator software or edge hardware itself.

```
Thu Jul 24 2025 08:09:11 GMT+0000 (Coordinated Universal Time), "[2025/07/24
10:09:11.201] TASK S: Failed request in device request=S05 device=ZIV0049222372
retry=1 error=\"node ZIV0049222372 is in active key pending failure\" (WARN)"
Thu Jul 24 2025 08:09:11 GMT+0000 (Coordinated Universal Time), "[2025/07/24
10:09:11.201] TASK S: Failed request in device request=S05 device=ZIV0049252082
retry=1 error=\"node ZIV0049252082 is temporary inactive\" (WARN)"
Thu Jul 24 2025 08:09:11 GMT+0000 (Coordinated Universal Time), "[2025/07/24
10:09:11.200] TASK S: Failed request in device request=S05 device=ZIV0049252215
retry=1 error=\"node ZIV0049252215 is in active key pending failure\" (WARN)"
Thu Jul 24 2025 08:09:11 GMT+0000 (Coordinated Universal Time), "[2025/07/24
10:09:11.200] TASK S: Failed request in device request=S05 device=ZIV0049271135
retry=1 error=\"node ZIV0049271135 is in active key pending failure\" (WARN)"
Thu Jul 24 2025 08:09:11 GMT+0000 (Coordinated Universal Time), "[2025/07/24
10:09:11.200] TASK S: Failed request in device request=S05 device=ZIV0049278365
retry=1 error=\"node ZIV0049278365 is in active key pending failure\" (WARN)"
Thu Jul 24 2025 08:09:11 GMT+0000 (Coordinated Universal Time), "[2025/07/24
10:09:11.200] TASK S: Failed request in device request=S05 device=ZIV0049284137
retry=1 error=\"node ZIV0049284137 is in active key pending failure\" (WARN)"
Thu Jul 24 2025 08:09:11 GMT+0000 (Coordinated Universal Time), "[2025/07/24
10:09:11.200] TASK S: Failed request in device request=S05 device=ZIV0049285782
retry=1 error=\"node ZIV0049285782 is in active key pending failure\" (WARN)"
Thu Jul 24 2025 08:09:11 GMT+0000 (Coordinated Universal Time), "[2025/07/24
10:09:11.200] TASK S: Failed request in device request=S05 device=ZIV0049288111
retry=1 error=\"node ZIV0049288111 is in active key pending failure\" (WARN)"
Thu Jul 24 2025 08:09:11 GMT+0000 (Coordinated Universal Time), "[2025/07/24
10:09:11.200] TASK S: Failed request in device request=S05 device=ZIV0049288889
retry=1 error=\"node ZIV0049288889 is in active key pending failure\" (WARN)"
Thu Jul 24 2025 08:09:11 GMT+0000 (Coordinated Universal Time), "[2025/07/24
10:09:11.200] TASK S: Failed request in device request=S05 device=ZIV0049296746
retry=1 error=\"node ZIV0049296746 is in active key pending failure\" (WARN)"
Thu Jul 24 2025 08:09:11 GMT+0000 (Coordinated Universal Time), "[2025/07/24
10:09:11.200] TASK S: Failed request in device request=S05 device=ZIV0049313695
retry=1 error=\"node ZIV0049313695 is in active key pending failure\" (WARN)"
Thu Jul 24 2025 08:09:11 GMT+0000 (Coordinated Universal Time), "[2025/07/24
10:09:11.200] TASK S: Failed request in device request=S05 device=ZIV0049321670
retry=1 error=\"node ZIV0049321670 is in active key pending failure\" (WARN)"
Thu Jul 24 2025 08:09:11 GMT+0000 (Coordinated Universal Time), "[2025/07/24
10:09:11.200] TASK S: Failed request in device request=S05 device=ZIV0049321989
retry=1 error=\"node ZIV0049321989 is in active key pending failure\" (WARN)"
Thu Jul 24 2025 08:09:11 GMT+0000 (Coordinated Universal Time), "[2025/07/24
10:09:11.200] TASK S: Failed request in device request=S05 device=ZIV0049322165
retry=1 error=\"node ZIV0049322165 is in active key pending failure\" (WARN)"
Thu Jul 24 2025 08:09:11 GMT+0000 (Coordinated Universal Time), "[2025/07/24
10:09:11.200] TASK S: Failed request in device request=S05 device=ZIV0049325222
retry=1 error=\"node ZIV0049325222 is in active key pending failure\" (WARN)"
```

Another aspect of scalability is the load placed on the edge node's hardware. It was observed that the virtual concentrator could handle up to its limit of 8 simultaneous DLMS/TCP sessions (connecting to 8 different CTs in parallel) without saturating the CPU or memory of the edge device. The resource consumption metrics of the VDC application during the stress test is depicted in Figure 22, This plot shows the edge node's CPU utilization, RAM usage, flash storage, persistent storage (PerStor), temporary storage (TmpStor), and PLC usage over time under the maximum load scenario (12 substations, thousands of meters).

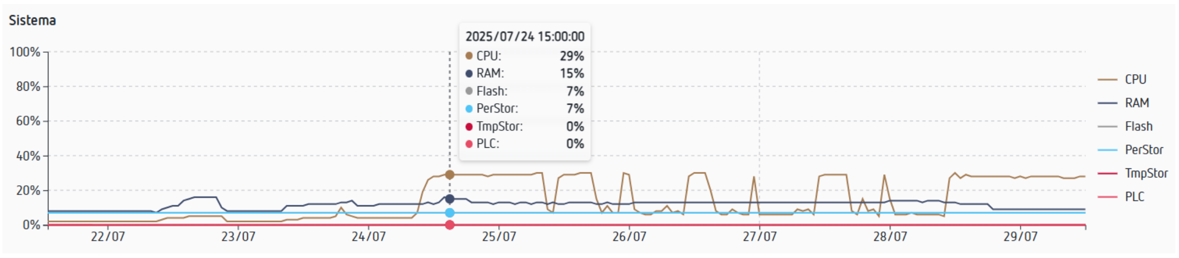


Figure 22: Resource consumption metrics of the VDC application during scalability tests, showing CPU, RAM, Flash, PerStor, TmpStor, and PLC usage over time

Additionally, Figure 23 provides the peak resource usage of the same parameters during the OLTC regulator use case. The measurements indicate that in both scenarios, the edge node's resource usage remained within acceptable limits (no critical resource utilzation occurred).

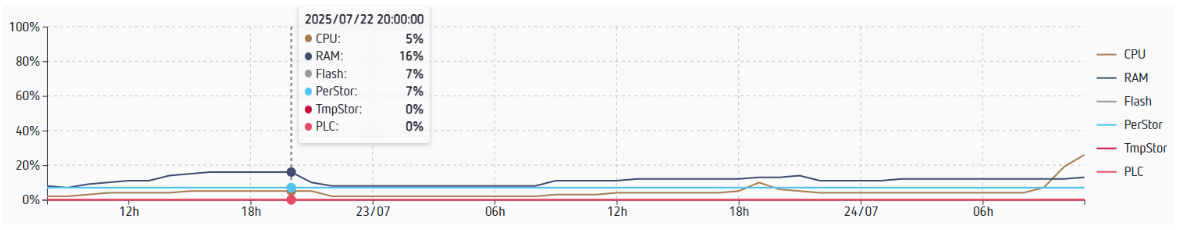


Figure 23:Resource consumption metrics of the VDC application during the use case testing, showing CPU, RAM, Flash, PerStor, TmpStor, and PLC usage over time

This confirms that the containerized VDC can scale to manage a high number of devices and parallel communications in real distribution network conditions, provided that external factors like security key management are handled appropriately.

Chapter 7. CONCLUSIONS AND FUTURE WORK

7.1 CONCLUSIONS

This section summarizes the key technical and practical contributions derived from the research and validation activities performed throughout this thesis. Each achievement reflects significant outcomes directly related to the implementation and validation of the Virtual Data Concentrator (VDC) and the associated edge computing platform.

Firstly, a technical comparative analysis was conducted between the Barbara edge platform currently adopted by Iberdrola i-DE and the previous Minsait solution. Both architectures were evaluated regarding their features, security approach, and suitability for distributed environments. The analysis concluded that Minsait aligns more closely with structured standardization and explicit cybersecurity requirements, whereas Barbara offers greater flexibility for third-party integration and innovation.

Secondly, the deployment and implementation of the Barbara platform and edge node were thoroughly documented, including detailed installation on industrial-grade hardware. This deployment demonstrated the technical feasibility of integrating a containerized edge computing solution within Iberdrola's operational environment and serves as a reproducible guideline for future deployments.

Thirdly, a modular edge computing architecture was proposed to assist the E4S alliance in the definition of standardized and interoperable models for secondary substations. As part of this contribution, the work defined and characterized key interfaces between the different architectural layers of the edge node, addressing gaps previously identified in the E4S reference framework. For Interface #6 (OS Layer – Node Agent), the use of Netlink sockets was identified as the most suitable option among the evaluated alternatives, given its compatibility with E4S requirements for efficient kernel-user space communication and low overhead. For Interface #7 (Node Agent – Applications), the analysis concluded that a

MQTT-driven architecture, potentially complemented with lightweight container orchestration APIs, offers promising advantages in terms of modularity, resilience to application restarts, and minimal resource consumption, which are key attributes for constrained substation environments. Finally, for Interface #8 (Applications – External Access or APIs), due to the anticipated heterogeneity and complexity of applications deployed at the edge often tailored to specific DSO needs, the adoption of a flexible and semantic-rich interface based on the use of WoT with CIM, is recommended to support scalability and interoperability in multi-vendor contexts.

Fourthly, the virtualized Data Concentrator solution was validated through comprehensive laboratory tests. These tests confirmed the VDC's capability to perform essential tasks such as configuration updates, connectivity management, measurement collection from smart meters, and scheduled task execution, thus verifying the technical viability of the virtualization approach.

Fifthly, a field-based edge use case for On-Load Tap Changer (OLTC) regulation and monitoring was developed and tested on field-deployed equipment, demonstrating real-time monitoring and autonomous control capabilities. The application successfully used real-time data from the VDC's metering interface to perform local voltage regulation calculations, illustrating new opportunities for edge computing in smart substations.

Sixthly, an economic impact analysis assessed the viability of deploying edge nodes at scale within the PRADA project. The cost-benefit analysis compared the virtualized approach against traditional solutions, highlighting the dependency of economic feasibility on regulatory recognition of software licensing as capital expenditure and the potential advantages of deploying multiple applications per edge node.

Finally, scalability testing was conducted on field-deployed equipment, confirming the VDC's ability to manage a significantly higher number of meters compared to typical deployments. Stress tests involving thousands of smart meters demonstrated stable resource usage and reliable operation, validating the VDC's readiness for large-scale implementation.

In conclusion, the project successfully demonstrated that containerized virtualization of Data Concentrators is technically viable, economically feasible under appropriate regulatory conditions, and scalable to meet future grid demands. These results provide a strong foundation for future deployments and developments in smart grid edge computing.

7.2 FUTURE WORK

The outcomes achieved in this thesis open several promising avenues for future development, particularly aimed at enhancing security, interoperability, and scalability of the virtualized edge platform:

- Secure Northbound Communications: Implement HTTPS with mutual TLS authentication for communications between the VDC and the remote management system (RMS), pending support from the vendor (Circutor). This will require proper configuration and integration of the VDC's digital certificate within Iberdrola's Public Key Infrastructure (PKI) to ensure trusted, encrypted data exchange.
- Virtualization of Additional Substation Agents: Extend the containerization architecture to other substation devices such as RTUs (Remote Terminal Units) and LVS (Low Voltage Supervisors). This would enable the edge platform to host multiple smart grid applications concurrently on a single node, consolidating functionalities like data concentration, control, and monitoring within one hardware unit.
- Deployment on High-Performance Hardware: Evaluate the feasibility of running the virtualized platform on over-provisioned substation hardware (for instance, deploying the VDC on a high-capacity RTU device). This assessment should examine multitasking performance and resource sharing when multiple containerized functions operate in parallel on the same host, ensuring that critical applications remain compatible and responsive under combined workloads.
- Decoupled Node Architecture: Further separate the base operating system from the edge agent software on the node. This decoupling would facilitate more flexible,

vendor-agnostic deployments by allowing the core OS platform to remain independent of any specific data concentrator or edge agent implementation, thereby easing future integration of different vendors' applications on standard hardware.

Chapter 8. REFERENCES

[1] Iberdrola, "www.iberdrola.com," [Online]. Available: https://www.i-de.es/glosario/smart-grids. [Accessed 27 May 2025].

[2] International Electrothecnical Commision, "IEC 61850 - Communication networks and systems for power utility automation," IEC, Geneva, 2013-2023.

[3] Barbara IoT SL, "Ficha de Producto: Barbara, la plataforma edge industrial cibersegura," 2022.

[4] Iberdrola: Negocio de Redes, "Distributed Processing Platform RFI evaluation Report," 2025.

[5] FUTURED, "CENTRO DE TRANSFORMACIÓN INTELIGENTE: FUNDAMENTOS PARA DISTRIBUCIÓN DE FUNCIONALIDADES," 2021.

[6] E4S Consortium, "E4S - DSO Requirements," 2025.

[7] i-DE - Grupo Iberdrola, "Technical Specification Data Concentrator," 2025.

[8] i-DE Grupo Iberdrola, "STG - DC Interface Specification Version 4.0," 2025.

[9] PRIME Alliance, "www.prime-alliance.org," [Online]. Available: https://www.prime-alliance.org/media/2020/04/PRIME-Spec_v1.4-20141031.pdf.

[10] W3C, "SOAP Specifications," [Online]. Available: https://www.w3.org/TR/2000/NOTE-SOAP-20000508/.

[11] D. T. M. D. P. D. B. G. N. D. Michael Metzer, "Introducing the grid impact score: an indicator for DER impact present and future grid sates," *CIRED 2024 Vienna Workshop,* 2024.

[12] Sagar Dayabhai, Peter Diamandis, "The role of virtualization in a smart-grid enabled substation automation system".

[13] P. E. O. A. D. L. V. V. a. G. E. Torres, "Virtualization in Substations. Technologies and applications," in *22nd International Conference on Renewable Energies and Power Quality* , Bilbao (Spain), 2024.

[14] T. K. I. K. I. H. N. C. L. P. T. P. P. P. Li Guoqing, "The Convergence of Container and Traditional Virtualization: Strengths and Limitations," *SN Computer Science,* vol. 4, 2023.

[15] H. A. D. R. H. R. T. T. C. Van Hoa Nguyen, "Containerized Iec 61850-Based Protection in Smart Grid – Lab Demonstration and Performance Assessment for Large Scale Deployment," 2024.

[16] J. a. G. C. a. X. Y. a. L. F. Li, "Edge-cloud Computing Systems for Smart Grid: State-of-the-art, Architecture, and Applications," *Journal of Modern Power Systems and Clean Energy,* vol. 10, no. 4, pp. 805-817, 2022.

[17] N. R. Pérez, "FINAL MASTER THESIS: Analysis of an edge-computing-based solution for local data processing at secondary substations," Madrid, 2020.

[18] Mariano Ortega de Mues, Daniel Seseña, Cesar Martinez Spessot, Marcos Carranza, Jorge Lang, "www.intel.com," 3 2 2020. [Online]. Available: https://www.intel.com/content/www/us/en/developer/articles/technical/edge-workload-consolidation-ewlc.html. [Accessed 04 06 2025].

[19] THE LINUX FOUNDATION, "SEAPATH - LF Energy," 2023. [Online]. Available: https://lfenergy.org/rte-deploys-lf-energy-seapath-for-virtual-protection-automation-and-control/.

[20] OpenAPI, "OpenAPI Specification v3.0.3," 20 02 2020. [Online]. Available: https://spec.openapis.org/oas/v3.0.3.html. [Accessed 06 06 2025].

[21] W3C, "Web of Things (WoT) Architecture 1.1," 05 12 2023. [Online]. Available: https://www.w3.org/TR/wot-architecture11/. [Accessed 06 06 2025].

[22] Indra - Minsait - Eduardo Jiménez Segado, "MiDE4S Libro de Trabajo CT Inteligente Funcionalidades de control, integración y de gestión," 2020.

[23] Barbara, "Despliegue Barbara SOLO en entorno de Iberdrola," Madrid, 2025.

[24] Linux, "The Linux Kernel 6.16.0-rc6," [Online]. Available: https://docs.kernel.org/userspace-api/netlink/intro.html.

[25] CNMC-Artículo 8. Retribución a la inversión en instalaciones cuya puesta en servicio ha sido posterior al 31 de diciembre de 2017., "www.boe.es," [Online]. Available: https://www.boe.es/diario_boe/txt.php?id=BOE-A-2019-18261.

[26] United Nations, "THE 17 GOALS - Sustainable Development Goals," [Online]. Available: https://sdgs.un.org/es/goals. [Accessed 12 06 2025].

[27] A. Sendin, J. Matanza and R. Ferrus, Smart Grid Telecommunications: Fundamentals and Technologies in the 5G Era, Wiley-IEEE Press books, 2021.

Chapter 9. APPENDICES

9.1 *APPENDIX I: CONFIGURATION AND FUNCTIONAL MEASUREMENT TESTS LOGS AND REPORTS*

Report S12 (Pre configuration)

```
<Report xmlns="http://stgdc/ws/S12" IdRpt="S12" IdPet="2155794473" Version="4.0">
    <Cnc Id="CIR4622301207">
        <S12 Fh="20250620093935078S">
            <INFO Mod="Compact DC" Af="2022" Te="concentrator" Vf="2.0.2"
Pro="ISDIP" Bat="100" Com="PLC" revConf="0_1" dateConf="20250620093935063S"
NomInstal="" CodInstal=""></INFO>
            <GENERAL MaxLogDepth="3000" StatisticsPeriod="60"
LogMask="0000111111111111" CheckSTG="N" TPLNumMax="10"></GENERAL>
            <NETWORK ipCom="200.0.0.1" DCPortWS="8080" ipMask="255.255.255.0"
ipGtw="200.0.0.100" ipDhcp="N" ipLoc="100.0.0.1"
ipMaskLoc="255.255.255.0"></NETWORK>
            <ACCESS AccInacTimeout="15" AccSimulMax="3" AuthIP="0.0.0.0"
AuthRetry="0" AuthRetryInterval="30" LdapCat1="" LdapCat2="" LdapCat3=""
LdapCat4="" LdapCat5="" LdapCat6="" LdapCat7="" LdapCat8="" TacacsEnable="N"
LdapBase="" LdapAuthentication="simple" LdapBindUser="" LdapStartTlsPolicy="1"
sshEnabled="Y" sshListenAddress="0.0.0.0:22" telnetEnabled="N"
telnetListenAddress="0.0.0.0:23" x509DefaultValidationPolicy="1"
tlsVersions="tls1_2,tls1_3" PkiUrl="" WebUIX509Authentication="N"
CertRenewEnabled="N" CertRenewRetryInterval="24" CertExpTime="720"
ipDns="0.0.0.0"></ACCESS>
            <TIME NtpMaxDeviation="30" ipNtp="100.200.253.16" ipNtp2="0.0.0.0"
toutNtp="60" nRetryNtp="3" tRetryNtp="5" timeSincroNtp="15"
timeZone="Europe/Madrid"></TIME>
            <DLMS DlmsEnable="Y" TimeDisconMeter="86400" RetryDisconMeter="40"
TimeRetryInterval="1800" ValuesCheckDelay="10" MaxOrderOutdate="600"
toutDCConfig="1440" TimeDelayRestart="60" SyncMeter="N" TimeDev="30"
TimeDevOver="3660"
MeterRegData="010000600100FF02010000600101FF02010000600102FF02"
TimeRegOver="43200" PLCTimeoutRM="180" PLCTimeoutF="7200" TimeOutMeterFwU="36000"
NumRetFwU="20" TimeFwURet="30" NumMeters="100" TimeSendReq="3600"></DLMS>
            <TASK MaxParallelOrders="10" ReadOnReconnection="N" RetryPerDev="N"
OddDaysSorting="Alphabetical" EvenDaysSorting="Alphabetical"
OddDaysSortingDirection="Forward" EvenDaysSortingDirection="Reverse"
ReportDepth="7" MaxTpTar="20" TaskSchedulerEnable="Y"></TASK>
            <MULTI_STG>
                <STG STGid="STG1" PortSTG="80" ReportFormat="0"
ipStg="100.200.253.35" StgProtocol="http" StgPath="/WS_STG/WS_STG.asmx"
nRetryWS="3" tRetryWS="30" toutWS="10" FtpProtocol="FTP" DestDirReport="/"
ipFtp="100.200.253.36" nRetryFtp="5" tRetryFtp="30" toutFtp="300"
```

```
FtpRandomDelay="0" FtpUserReport="CONFIDENTIAL" ipFtpDCUpg="100.200.253.36"
UserFtpDCUpg="CONFIDENTIAL" ipFtpMeterUpg="100.200.253.36"
UserFtpMeterUpg="CONFIDENTIAL"></STG>
                <STG STGid="STG2" PortSTG="8080" ReportFormat="0"
ipStg="100.0.0.100" StgProtocol="http" StgPath="StgPath=/WS_STG/WS_STG.asmx"
nRetryWS="3" tRetryWS="30" toutWS="10" FtpProtocol="FTP" DestDirReport="/"
ipFtp="100.0.0.100" nRetryFtp="5" tRetryFtp="300" toutFtp="10" FtpRandomDelay="0"
FtpUserReport="SC_TGU" ipFtpDCUpg="100.0.0.100" UserFtpDCUpg="SC_TGU"
ipFtpMeterUpg="100.0.0.100" UserFtpMeterUpg="SC_TGU"></STG>
                <STG STGid="STG3" PortSTG="80" ReportFormat="0"
ipStg="100.0.0.100" StgProtocol="http" StgPath="/WS_STG/WS_STG.asmx" nRetryWS="3"
tRetryWS="30" toutWS="10" FtpProtocol="FTP" DestDirReport="/" ipFtp="100.0.0.100"
nRetryFtp="5" tRetryFtp="300" toutFtp="10" FtpRandomDelay="600"
FtpUserReport="SC_TGU" ipFtpDCUpg="100.0.0.100" UserFtpDCUpg="SC_TGU"
ipFtpMeterUpg="100.0.0.100" UserFtpMeterUpg="SC_TGU"></STG>
            </MULTI_STG>
            <EVENTS
CfgDCEvMask1="FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF"
CfgDCEvMask2="FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF"
CfgDCEvMask3="FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF"
CfgDCEvMask4="FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF"
CfgDCEvMask5="FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF"
SpontSTGDest="1"
SponDCEvMask1="0000000000000000000000000000000000000000000000000000000000000002"
SponDCEvMask2="0000000000000000000000000000000000000000000000000000000000000000"
SponDCEvMask3="0000000000000000000000000000000000000000000000000000000000000000"
SponDCEvMask4="0000000000000000000000000000000000000000000000000000000000000000"
SponDCEvMask5="000000000000000000000000000000000000000000000000000000000000000A">
</EVENTS>
            <DLMSovTCP></DLMSovTCP>
            <SNMP SnmpEnable="N" SnmpVersion="2" SnmpPublic="public"
SnmpPrivate="private" TrapCommunity="public" TrapAddress="0.0.0.0"
Snmp3SecLevel="authPriv" Snmp3SecUser=""></SNMP>
            <FTPcycles FtpCyclesProtocol="FTP" ipFtpCycles="100.200.253.15"
UserFtpCycles="trazas" DestDirCycles="/" PrefixCycles=""></FTPcycles>
            <Other>
                <Parameter Group="DLMS" Key="CaptureFramesFilter"
Value=".+"></Parameter>
                <Parameter Group="STG" Key="Version" Value="4.0"></Parameter>
                <Parameter Group="STG" Key="ResetMsg" Value="false"></Parameter>
                <Parameter Group="ACCESS" Key="DHCPIdentifier"
Value=""></Parameter>
                <Parameter Group="DLMS" Key="S26Content"
Value="Pimp;AIa;L1v"></Parameter>
                <Parameter Group="DLMS" Key="RetryDisconnectMeterOrder"
Value="0"></Parameter>
                <Parameter Group="DLMS" Key="TimeRetryIntervalOrder"
Value="0"></Parameter>
                <Parameter Group="STG" Key="ResetRandomDelay"
Value="0"></Parameter>
                <Parameter Group="STG" Key="Priority" Value="true"></Parameter>
            </Other>
        </S12>
```

```
    </Cnc>
</Report>
```

Order B07

```
    <Order xmlns:i="http://www.w3.org/2001/XMLSchema-instance"
xmlns="urn:stgdc/dc/ws">
      <IdPet>2155794474</IdPet>
      <Format>0</Format>
      <Order>
        <Order xmlns="http://stgdc/ws/B07" IdReq="B07" IdPet="2155794474"
Version="4.0">
          <Cnc Id="CIR4622301207">
            <B07>
              <MULTI_STG>
                <STG STGid="STG1" PortSTG="80" ReportFormat="0"
ipStg="100.200.253.35" StgProtocol="http" StgPath="/WS_STG/WS_STGv4.asmx"
nRetryWS="3" tRetryWS="30" toutWS="10" ipFtp="100.200.253.36" nRetryFtp="5"
tRetryFtp="30" toutFtp="300" FtpRandomDelay="0" FtpUserReport="CONFIDENTIAL"
FtpPwdReport="CONFIDENTIAL" ipFtpDCUpg="100.200.253.36"
UserFtpDCUpg="CONFIDENTIAL" PwdFtpDCUpg="CONFIDENTIAL"
ipFtpMeterUpg="100.200.253.36" UserFtpMeterUpg="CONFIDENTIAL"
PwdFtpMeterUpg="CONFIDENTIAL"/>

                <STG STGid="STG2" PortSTG="8080" ReportFormat="0"
ipStg="100.0.0.100" StgProtocol="http"
StgPath="StgPath=&quot;/WS_STG/WS_STG.asmx" nRetryWS="3" tRetryWS="30"
toutWS="10" ipFtp="100.0.0.100" nRetryFtp="5" tRetryFtp="300" toutFtp="10"
FtpRandomDelay="0" FtpUserReport="SC_TGU" FtpPwdReport="SC_TGU"
ipFtpDCUpg="100.0.0.100" UserFtpDCUpg="SC_TGU" PwdFtpDCUpg="SC_TGU"
ipFtpMeterUpg="100.0.0.100" UserFtpMeterUpg="SC_TGU" PwdFtpMeterUpg="SC_TGU"/>
              </MULTI_STG>
              <DLMSovTCP>
                <DlmsC DlmsC_id="1" DlmsC_ip_Addr="200.95.222.90" DlmsC_Type="bn"
DlmsC_Tcp_Port="4059" DlmsC_Descr="NB_TGU_CIR_LABO#1"/>
              </DLMSovTCP>
            </B07>
          </Cnc>
        </Order>
      </Order>
      <Fini>20250620100002526S</Fini>
      <Ffin>20250620100212526S</Ffin>
      <Priority>0</Priority>
    </Order>
```

Report S12 (Post configuration)

```
<Report xmlns="http://stgdc/ws/S12" IdRpt="S12" IdPet="0" Version="4.0">
    <Cnc Id="CIR4622301207">
        <S12 Fh="20250718115335083S">
```

```
            <INFO Mod="Compact DC" Af="2022" Te="concentrator" Vf="2.0.3"
Pro="ISDIP" Bat="100" Com="PLC" revConf="0_37" dateConf="20250717133418397S"
NomInstal="" CodInstal=""></INFO>
            <GENERAL MaxLogDepth="3000" StatisticsPeriod="60"
LogMask="0000111111111111" CheckSTG="N" TPLNumMax="10"></GENERAL>
            <NETWORK ipCom="200.0.0.1" DCPortWS="8080" ipMask="255.255.255.0"
ipGtw="200.0.0.100" ipDhcp="N" ipLoc="100.0.0.1"
ipMaskLoc="255.255.255.0"></NETWORK>
            <ACCESS AccInacTimeout="15" AccSimulMax="3" AuthIP="0.0.0.0"
AuthRetry="0" AuthRetryInterval="30" LdapCat1="" LdapCat2="" LdapCat3=""
LdapCat4="" LdapCat5="" LdapCat6="" LdapCat7="" LdapCat8="" TacacsEnable="N"
LdapBase="" LdapAuthentication="simple" LdapBindUser="" LdapStartTlsPolicy="1"
sshEnabled="Y" sshListenAddress="0.0.0.0:22" telnetEnabled="N"
telnetListenAddress="0.0.0.0:23" x509DefaultValidationPolicy="1"
tlsVersions="tls1_2,tls1_3" PkiUrl="" WebUIX509Authentication="N"
CertRenewEnabled="N" CertRenewRetryInterval="24" CertExpTime="720"
ipDns="0.0.0.0"></ACCESS>
            <TIME NtpMaxDeviation="30" ipNtp="100.200.253.16" ipNtp2="0.0.0.0"
toutNtp="60" nRetryNtp="3" tRetryNtp="5" timeSincroNtp="15"
timeZone="Europe/Madrid"></TIME>
            <DLMS DlmsEnable="Y" TimeDisconMeter="86400" RetryDisconMeter="40"
TimeRetryInterval="1800" ValuesCheckDelay="10" MaxOrderOutdate="600"
toutDCConfig="1440" TimeDelayRestart="60" SyncMeter="N" TimeDev="30"
TimeDevOver="3660"
MeterRegData="010000600100FF02010000600101FF02010000600102FF02"
TimeRegOver="43200" PLCTimeoutRM="180" PLCTimeoutF="7200" TimeOutMeterFwU="36000"
NumRetFwU="20" TimeFwURet="30" NumMeters="100" TimeSendReq="3600"></DLMS>
            <TASK MaxParallelOrders="10" ReadOnReconnection="N" RetryPerDev="N"
OddDaysSorting="Alphabetical" EvenDaysSorting="Alphabetical"
OddDaysSortingDirection="Forward" EvenDaysSortingDirection="Reverse"
ReportDepth="7" MaxTpTar="20" TaskSchedulerEnable="Y"></TASK>
            <MULTI_STG>
                <STG STGid="STG1" PortSTG="80" ReportFormat="0"
ipStg="100.200.253.35" StgProtocol="http" StgPath="/WS_STG/WS_STGv4.asmx"
nRetryWS="3" tRetryWS="30" toutWS="10" FtpProtocol="FTP" DestDirReport="/"
ipFtp="100.200.253.36" nRetryFtp="5" tRetryFtp="30" toutFtp="300"
FtpRandomDelay="0" FtpUserReport="CONFIDENTIAL" ipFtpDCUpg="100.200.253.36"
UserFtpDCUpg="CONFIDENTIAL" ipFtpMeterUpg="100.200.253.36"
UserFtpMeterUpg="CONFIDENTIAL"></STG>
                <STG STGid="STG2" PortSTG="8080" ReportFormat="0"
ipStg="100.0.0.100" StgProtocol="http" StgPath="StgPath=/WS_STG/WS_STG.asmx"
nRetryWS="3" tRetryWS="30" toutWS="10" FtpProtocol="FTP" DestDirReport="/"
ipFtp="100.0.0.100" nRetryFtp="5" tRetryFtp="300" toutFtp="10" FtpRandomDelay="0"
FtpUserReport="SC_TGU" ipFtpDCUpg="100.0.0.100" UserFtpDCUpg="SC_TGU"
ipFtpMeterUpg="100.0.0.100" UserFtpMeterUpg="SC_TGU"></STG>
                <STG STGid="STG3" PortSTG="80" ReportFormat="0"
ipStg="100.0.0.100" StgProtocol="http" StgPath="/WS_STG/WS_STG.asmx" nRetryWS="3"
tRetryWS="30" toutWS="10" FtpProtocol="FTP" DestDirReport="/" ipFtp="100.0.0.100"
nRetryFtp="5" tRetryFtp="300" toutFtp="10" FtpRandomDelay="600"
FtpUserReport="SC_TGU" ipFtpDCUpg="100.0.0.100" UserFtpDCUpg="SC_TGU"
ipFtpMeterUpg="100.0.0.100" UserFtpMeterUpg="SC_TGU"></STG>
            </MULTI_STG>
            <EVENTS
CfgDCEvMask1="FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF"
CfgDCEvMask2="FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF"
CfgDCEvMask3="FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF"
CfgDCEvMask4="FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF"
CfgDCEvMask5="FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF"
SpontSTGDest="1"
SponDCEvMask1="0000000000000000000000000000000000000000000000000000000000000002"
SponDCEvMask2="0000000000000000000000000000000000000000000000000000000000000000"
SponDCEvMask3="0000000000000000000000000000000000000000000000000000000000000000"
SponDCEvMask4="0000000000000000000000000000000000000000000000000000000000000000"
SponDCEvMask5="000000000000000000000000000000000000000000000000000000000000000A">
</EVENTS>
            <DLMSovTCP>
                <DlmsC DlmsC_id="1" DlmsC_ip_Addr="200.95.333.18" DlmsC_Type="bn"
DlmsC_Tcp_Port="4059" DlmsC_Descr="NB_TGU_ZIV_Karpo"></DlmsC>
                <DlmsC DlmsC_id="2" DlmsC_ip_Addr="200.95.222.90" DlmsC_Type="bn"
DlmsC_Tcp_Port="4059" DlmsC_Descr="NB_TGU_CIR_LABO#1"></DlmsC>
            </DLMSovTCP>
            <SNMP SnmpEnable="N" SnmpVersion="2" SnmpPublic="public"
SnmpPrivate="private" TrapCommunity="public" TrapAddress="0.0.0.0"
Snmp3SecLevel="authPriv" Snmp3SecUser=""></SNMP>
            <FTPcycles FtpCyclesProtocol="FTP" ipFtpCycles="100.200.253.15"
UserFtpCycles="trazas" DestDirCycles="/" PrefixCycles=""></FTPcycles>
            <Other>
                <Parameter Group="DLMS" Key="CaptureFramesFilter"
Value=".+"></Parameter>
                <Parameter Group="STG" Key="Version" Value="4.0"></Parameter>
                <Parameter Group="STG" Key="ResetMsg" Value="false"></Parameter>
                <Parameter Group="ACCESS" Key="DHCPIdentifier"
Value=""></Parameter>
                <Parameter Group="DLMS" Key="S26Content"
Value="Pimp;AIa;L1v"></Parameter>
                <Parameter Group="DLMS" Key="RetryDisconnectMeterOrder"
Value="0"></Parameter>
                <Parameter Group="DLMS" Key="TimeRetryIntervalOrder"
Value="0"></Parameter>
                <Parameter Group="STG" Key="ResetRandomDelay"
Value="0"></Parameter>
                <Parameter Group="STG" Key="Priority" Value="true"></Parameter>
            </Other>
        </S12>
    </Cnc>
</Report>
```

Report S24

```
<Report xmlns="http://stgdc/ws/S24" IdRpt="S24" IdPet="0" Version="4.0">
    <Cnc Id="CIR4622301207">
        <S24 Fh="20250630152143708S">
            <Meter MeterId="AUX1023030017" ComStatus="3"
Date="20250627142500428S" Active="Y"></Meter>
```

```
            <Meter MeterId="ELS0032114924" ComStatus="2"
Date="20250627142457701S" Active="Y"></Meter>
            <Meter MeterId="ITE0141020666" ComStatus="2"
Date="20250627142505351S" Active="Y"></Meter>
            <Meter MeterId="ORB0000803879" ComStatus="2"
Date="20250627142506902S" Active="Y"></Meter>
            <Meter MeterId="SAG0125322720" ComStatus="2"
Date="20250627144414583S" Active="Y"></Meter>
            <Meter MeterId="SAG0186301287" ComStatus="3"
Date="20250627142448082S" Active="Y"></Meter>
            <Meter MeterId="SOG0000502076" ComStatus="3"
Date="20250627142450824S" Active="Y"></Meter>
            <Meter MeterId="ZIV0039808892" ComStatus="3"
Date="20250627142456368S" Active="Y"></Meter>
            <Meter MeterId="ZIV0044603022" ComStatus="5"
Date="20250627142355881S" Active="Y"></Meter>
        </S24>
    </Cnc>
</Report>
```

Report G24

```
<Report xmlns="http://stgdc/ws/G24" IdRpt="G24" IdPet="0" Version="4.0">
    <Cnc Id="CIR4622301207">
        <G24 Fh="20250630152240937S">
            <Meter MeterId="AUX1023030017" MAC="50:0F:59:B7:8E:F1" ComStatus="2"
SecurityStatus="3" LastCom="20250627142500428S" Mod="AB" Te="contador" Vf="V1365"
VPrime="V9372" Atimeperc="100.00" Nchanges="0" NBip="200.95.222.90"></Meter>
            <Meter MeterId="ELS0032114924" MAC="00:23:7E:FC:28:22" ComStatus="2"
SecurityStatus="2" LastCom="20250627142457701S" Mod="CR" Te="contador" Vf="V0117"
VPrime="6.28.2.13" Atimeperc="100.00" Nchanges="0" NBip="200.95.222.90"></Meter>
            <Meter MeterId="ITE0141020666" MAC="00:07:81:02:83:5D" ComStatus="2"
SecurityStatus="2" LastCom="20250627142505351S" Mod="KD" Te="contador" Vf="V0214"
VPrime="7.7.1.0" Atimeperc="100.00" Nchanges="0" NBip="200.95.222.90"></Meter>
            <Meter MeterId="ORB0000803879" MAC="70:64:17:1C:44:27" ComStatus="2"
SecurityStatus="2" LastCom="20250627142506902S" Mod="CM" Te="contador" Vf="V0011"
VPrime="01.03.09.06" Atimeperc="100.00" Nchanges="0"
NBip="200.95.222.90"></Meter>
            <Meter MeterId="SAG0125322720" MAC="C0:AC:54:FE:4E:00" ComStatus="2"
SecurityStatus="2" LastCom="20250627144414583S" Mod="BT" Te="contador" Vf="V0319"
VPrime="06.35.00.20" Atimeperc="100.00" Nchanges="0"
NBip="200.95.222.90"></Meter>
            <Meter MeterId="SAG0186301287" MAC="A0:1B:29:88:E2:C9" ComStatus="2"
SecurityStatus="3" LastCom="20250627142448082S" Mod="CA" Te="Tipo4MDPLC"
Vf="V0455" VPrime="06.35.00.20" Atimeperc="100.00" Nchanges="0"
NBip="200.95.222.90"></Meter>
            <Meter MeterId="SOG0000502076" MAC="D4:8F:AA:09:81:EA" ComStatus="2"
SecurityStatus="3" LastCom="20250627142450824S" Mod="BN" Te="contador" Vf="V0400"
VPrime="010305020FU" Atimeperc="100.00" Nchanges="0"
NBip="200.95.222.90"></Meter>
            <Meter MeterId="ZIV0039808892" MAC="40:40:22:5F:6F:7C" ComStatus="2"
SecurityStatus="3" LastCom="20250627142456368S" Mod="NM" Te="contador" Vf="VK607"
VPrime="V2101" Atimeperc="100.00" Nchanges="0" NBip="200.95.222.90"></Meter>
```

```
            <Meter MeterId="ZIV0044603022" MAC="40:40:22:A8:96:8E" ComStatus="2"
SecurityStatus="5" LastCom="20250627142355881S" Mod="" Te="" Vf="" VPrime=""
Atimeperc="100.00" Nchanges="0" NBip="200.95.222.90"></Meter>
        </G24>
    </Cnc>
</Report>
```

9.2 *APPENDIX II: SCHEDULED TASKS TESTS LOGS AND REPORTS*

Logs S05

```
[2025/07/01 00:10:00.769] TASKS: Activated task task=1 (INFO)
[2025/07/01 00:10:00.779] TASKS: Activated task task=2 (INFO)
[2025/07/01 00:10:11.410] TASKS: Finished request in device request=S02
device=AUX1023030017 (INFO)
[2025/07/01 00:10:18.376] TASKS: Finished request in device request=S02
device=ELS0032114924 (INFO)
[2025/07/01 00:10:21.800] TASKS: Finished request in device request=S02
device=ITE0141020666 (INFO)
[2025/07/01 00:10:31.612] TASKS: Finished request in device request=S02
device=ORB0000803879 (INFO)
[2025/07/01 00:10:37.615] TASKS: Finished request in device request=S02
device=SAG0125322720 (INFO)
[2025/07/01 00:11:02.598] TASKS: Finished request in device request=S02
device=SAG0186301287 (INFO)
[2025/07/01 00:11:16.082] TASKS: Finished request in device request=S02
device=SOG0000502076 (INFO)
[2025/07/01 00:11:21.160] TASKS: Finished request in device request=S02
device=ZIV0039808892 (INFO)
[2025/07/01 00:11:21.160] TASKS: Failed request in device request=S02
device=ZIV0044603022 retry=1 error="node ZIV0044603022 is in active key pending
failure" (WARN)
[2025/07/01 00:11:21.160] TASKS: Finished retry round request=S02 retry=1 (INFO)
[2025/07/01 00:11:28.636] TASKS: Finished request in device request=S05
device=AUX1023030017 (INFO)
[2025/07/01 00:11:30.990] TASKS: Finished request in device request=S05
device=ELS0032114924 (INFO)
[2025/07/01 00:11:32.836] TASKS: Finished request in device request=S05
device=ITE0141020666 (INFO)
[2025/07/01 00:11:36.282] TASKS: Finished request in device request=S05
device=ORB0000803879 (INFO)
[2025/07/01 00:11:39.669] TASKS: Finished request in device request=S05
device=SAG0125322720 (INFO)
[2025/07/01 00:11:43.950] TASKS: Finished request in device request=S05
device=SAG0186301287 (INFO)
[2025/07/01 00:11:46.500] TASKS: Finished request in device request=S05
device=SOG0000502076 (INFO)
[2025/07/01 00:11:49.356] TASKS: Finished request in device request=S05
device=ZIV0039808892 (INFO)
```

```
[2025/07/01 00:11:49.356] TASKS: Failed request in device request=S05
device=ZIV0044603022 retry=1 error="node ZIV0044603022 is in active key pending
failure" (WARN)
[2025/07/01 00:11:49.356] TASKS: Finished retry round request=S05 retry=1 (INFO)
```

Report S02

```
<Report xmlns="http://stgdc/ws/S02" IdRpt="S02" IdPet="0" Version="4.0">
    <Cnc Id="CIR4622301207">
        <Cnt Id="AUX1023030017" Magn="1">
            <S02 Fh="20250704010000000S" Bc="02" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704020000000S" Bc="02" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704030000000S" Bc="02" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704040000000S" Bc="02" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704050000000S" Bc="02" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704060000000S" Bc="02" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704070000000S" Bc="02" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704080000000S" Bc="02" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704090000000S" Bc="02" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704100000000S" Bc="02" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704110000000S" Bc="02" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704120000000S" Bc="02" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704130000000S" Bc="06" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704140000000S" Bc="02" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704150000000S" Bc="02" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704160000000S" Bc="02" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704170000000S" Bc="02" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704180000000S" Bc="02" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704190000000S" Bc="02" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704200000000S" Bc="02" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
```

```
            <S02 Fh="20250704210000000S" Bc="02" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704220000000S" Bc="02" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704230000000S" Bc="12" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250705000000000S" Bc="02" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
        </Cnt>
        <Cnt Id="ELS0032114924" Magn="1">
            <S02 Fh="20250704010000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704020000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704030000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704040000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704050000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704060000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704070000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704080000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704090000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704100000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704110000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704120000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704130000000S" Bc="02" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704140000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704150000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704160000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704170000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704180000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704190000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704200000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704210000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
```

```
            <S02 Fh="20250704220000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704230000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250705000000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
        </Cnt>
        <Cnt Id="ITE0141020666" Magn="1">
            <S02 Fh="20250704010000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704020000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704030000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704040000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704050000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704060000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704070000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704080000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704090000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704100000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704110000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704120000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704130000000S" Bc="04" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704140000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704150000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704160000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704170000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704180000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704190000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704200000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704210000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704220000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
```

```
            <S02 Fh="20250704230000000S" Bc="10" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250705000000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
        </Cnt>
        <Cnt Id="ORB0000803879" Magn="1">
            <S02 Fh="20250704010000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704020000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704030000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704040000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704050000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704060000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704070000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704080000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704090000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704100000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704110000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704120000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704130000000S" Bc="02" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704140000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704150000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704160000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704170000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704180000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704190000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704200000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704210000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704220000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704230000000S" Bc="10" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
```

```
            <S02 Fh="20250705000000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
        </Cnt>
        <Cnt Id="SAG0125322720" Magn="1">
            <S02 Fh="20250704010000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704020000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704030000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704040000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704050000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704060000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704070000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704080000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704090000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704100000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704110000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704120000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704130000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704140000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704150000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704160000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704170000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704180000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704190000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704200000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704210000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704220000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704230000000S" Bc="18" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250705000000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
        </Cnt>
        <Cnt Id="SAG0186301287" Magn="1">
            <S02 Fh="20250704010000000S" Bc="82" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704020000000S" Bc="82" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704030000000S" Bc="82" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704040000000S" Bc="82" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704050000000S" Bc="82" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704060000000S" Bc="82" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704070000000S" Bc="82" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704080000000S" Bc="82" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704090000000S" Bc="82" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704100000000S" Bc="82" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704110000000S" Bc="82" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704120000000S" Bc="82" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704130000000S" Bc="82" AI="56" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704140000000S" Bc="82" AI="60" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704150000000S" Bc="82" AI="61" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704160000000S" Bc="82" AI="61" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704170000000S" Bc="82" AI="62" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704180000000S" Bc="82" AI="61" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704190000000S" Bc="82" AI="61" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704200000000S" Bc="82" AI="60" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704210000000S" Bc="82" AI="62" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704220000000S" Bc="82" AI="61" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704230000000S" Bc="9A" AI="61" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250705000000000S" Bc="82" AI="61" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
        </Cnt>
        <Cnt Id="SOG0000502076" Magn="1">
            <S02 Fh="20250704010000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
```

```
            <S02 Fh="20250704020000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704030000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704040000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704050000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704060000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704070000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704080000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704090000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704100000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704110000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704120000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704130000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704140000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704150000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704160000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704170000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704180000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704190000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704200000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704210000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704220000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704230000000S" Bc="10" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250705000000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
        </Cnt>
        <Cnt Id="ZIV0039808892" Magn="1">
            <S02 Fh="20250704010000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704020000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
```

```
            <S02 Fh="20250704030000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704040000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704050000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704060000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704070000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704080000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704090000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704100000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704110000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704120000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704130000000S" Bc="02" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704140000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704150000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704160000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704170000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704180000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704190000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704200000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704210000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704220000000S" Bc="18" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250704230000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
            <S02 Fh="20250705000000000S" Bc="00" AI="0" AE="0" R1="0" R2="0"
R3="0" R4="0"></S02>
        </Cnt>
    </Cnc>
</Report>
```

Logs S04

```
[2025/07/01 01:10:00.769] TASKS: Activated task task=3 (INFO)
[2025/07/01 01:10:00.779] TASKS: Data sent due TimeSendReq task=1 request=S05
(INFO)
[2025/07/01 01:10:01.769] TASKS: Data sent due TimeSendReq task=2 request=S02
(INFO)
[2025/07/01 01:10:04.402] TASKS: Finished request in device request=S04
device=AUX1023030017 (INFO)
[2025/07/01 01:10:07.322] TASKS: Finished request in device request=S04
device=ELS0032114924 (INFO)
[2025/07/01 01:10:10.350] TASKS: Finished request in device request=S04
device=ITE0141020666 (INFO)
[2025/07/01 01:10:16.692] TASKS: Finished request in device request=S04
device=ORB0000803879 (INFO)
[2025/07/01 01:10:20.663] TASKS: Finished request in device request=S04
device=SAG0125322720 (INFO)
[2025/07/01 01:10:25.551] TASKS: Finished request in device request=S04
device=SAG0186301287 (INFO)
[2025/07/01 01:10:29.587] TASKS: Finished request in device request=S04
device=SOG0000502076 (INFO)
[2025/07/01 01:10:34.751] TASKS: Finished request in device request=S04
device=ZIV0039808892 (INFO)
[2025/07/01 01:10:34.751] TASKS: Failed request in device request=S04
device=ZIV0044603022 retry=1 error="node ZIV0044603022 is in active key pending
failure" (WARN)
[2025/07/01 01:10:34.751] TASKS: Finished retry round request=S04 retry=1 (INFO)
[2025/07/01 01:11:22.769] TASKS: Failed request in device request=S02
device=ZIV0044603022 retry=3 error="node ZIV0044603022 is in active key pending
failure" (WARN)
[2025/07/01 01:11:22.769] TASKS: Finished retry round request=S02 retry=3 (INFO)
[2025/07/01 01:11:50.769] TASKS: Failed request in device request=S05
device=ZIV0044603022 retry=3 error="node ZIV0044603022 is in active key pending
failure" (WARN)
[2025/07/01 01:11:50.769] TASKS: Finished retry round request=S05 retry=3 (INFO)
```

Logs S09

```
[2025/07/01 03:00:00.769] TASKS: Activated task task=4 (INFO)
[2025/07/01 03:00:12.469] TASKS: Finished request in device request=S09
device=AUX1023030017 (INFO)
[2025/07/01 03:00:16.807] TASKS: Finished request in device request=S09
device=ELS0032114924 (INFO)
[2025/07/01 03:00:21.670] TASKS: Finished request in device request=S09
device=ITE0141020666 (INFO)
[2025/07/01 03:00:31.092] TASKS: Failed request in device request=T07
device=ZIV0044603022 retry=11 error="node ZIV0044603022 is in active key pending
failure" (WARN)
[2025/07/01 03:00:31.092] TASKS: Finished retry round request=T07 retry=11 (INFO)
[2025/07/01 03:00:31.096] TASKS: Finished request in device request=S09
device=ORB0000803879 (INFO)
```

```
[2025/07/01 03:00:35.980] TASKS: Finished request in device request=S09
device=SAG0125322720 (INFO)
[2025/07/01 03:00:42.820] TASKS: Finished request in device request=S09
device=SAG0186301287 (INFO)
[2025/07/01 03:00:49.386] TASKS: Finished request in device request=S09
device=SOG0000502076 (INFO)
[2025/07/01 03:00:56.199] TASKS: Finished request in device request=S09
device=ZIV0039808892 (INFO)
[2025/07/01 03:00:56.199] TASKS: Failed request in device request=S09
device=ZIV0044603022 retry=1 error="node ZIV0044603022 is in active key pending
failure" (WARN)
[2025/07/01 03:00:56.199] TASKS: Finished retry round request=S09 retry=1 (INFO)
```

Logs T01

```
[2025/07/06 15:00:00.769] TASKS: Activated task task=5 (INFO)
[2025/07/06 15:00:00.792] TASKS: Finished request task=5 request=T01 (INFO)
[2025/07/06 15:00:00.800] TASKS: Finished task task=5 (INFO)
```

Logs T07

```
[2025/06/30 22:00:00.769] TASKS: Activated task task=6 (INFO)
[2025/06/30 22:00:00.776] TASKS: Failed request in device request=T07
device=ZIV0044603022 retry=1 error="node ZIV0044603022 is in active key pending
failure" (WARN)
[2025/06/30 22:00:03.121] TASKS: Finished request in device request=T07
device=ZIV0039808892 (INFO)
[2025/06/30 22:00:04.293] TASKS: Finished request in device request=T07
device=SOG0000502076 (INFO)
[2025/06/30 22:00:05.252] TASKS: Finished request in device request=T07
device=SAG0186301287 (INFO)
[2025/06/30 22:00:07.469] TASKS: Finished request in device request=T07
device=SAG0125322720 (INFO)
[2025/06/30 22:00:11.969] TASKS: Finished request in device request=T07
device=ORB0000803879 (INFO)
[2025/06/30 22:00:13.727] TASKS: Finished request in device request=T07
device=ITE0141020666 (INFO)
[2025/06/30 22:00:14.609] TASKS: Finished request in device request=T07
device=ELS0032114924 (INFO)
[2025/06/30 22:00:19.325] TASKS: Finished request in device request=T07
device=AUX1023030017 (INFO)
[2025/06/30 22:00:19.325] TASKS: Finished retry round request=T07 retry=1 (INFO)
[2025/06/30 22:30:19.769] TASKS: Failed request in device request=T07
device=ZIV0044603022 retry=2 error="node ZIV0044603022 is in active key pending
failure" (WARN)
[2025/06/30 22:30:19.769] TASKS: Finished retry round request=T07 retry=2 (INFO)
```

Logs S17

```
[2025/06/30 22:50:00.769] TASKS: Activated task task=7 (INFO)
[2025/06/30 22:50:01.781] TASKS: Finished request task=7 request=S17 (INFO)
```

```
[2025/06/30 22:50:01.789] TASKS: Finished task task=7 (INFO)
```

Report S24

```
<Report xmlns="http://stgdc/ws/S24" IdRpt="S24" IdPet="0" Version="4.0">
    <Cnc Id="CIR4622301207">
        <S24 Fh="20250630224000775S">
            <Meter MeterId="AUX1023030017" ComStatus="3"
Date="20250630220018168S" Active="Y"></Meter>
            <Meter MeterId="ELS0032114924" ComStatus="2"
Date="20250630220014275S" Active="Y"></Meter>
            <Meter MeterId="ITE0141020666" ComStatus="2"
Date="20250630220012664S" Active="Y"></Meter>
            <Meter MeterId="ORB0000803879" ComStatus="2"
Date="20250630220009263S" Active="Y"></Meter>
            <Meter MeterId="SAG0125322720" ComStatus="2"
Date="20250630220005791S" Active="Y"></Meter>
            <Meter MeterId="SAG0186301287" ComStatus="3"
Date="20250630220004656S" Active="Y"></Meter>
            <Meter MeterId="SOG0000502076" ComStatus="3"
Date="20250630220003487S" Active="Y"></Meter>
            <Meter MeterId="ZIV0039808892" ComStatus="3"
Date="20250630220001049S" Active="Y"></Meter>
            <Meter MeterId="ZIV0044603022" ComStatus="5"
Date="20250627142355881S" Active="Y"></Meter>
        </S24>
    </Cnc>
</Report>
```

Report S11

```
<Report xmlns="http://stgdc/ws/S11" IdRpt="S11" IdPet="0" Version="4.0">
    <Cnc Id="CIR4622301207">
        <S11 Fh="20250630224000775S">
            <MacSNA macSNA="" macBeaconsPerFrame="0" macState="" macSCPLength="0"
macNodeHierarchyLevel="0" macBeaconSlotCount="0" macBeaconTxSlot="0"
macBeaconTxFrequency="0" macCSMAChBusyCount="0"></MacSNA>
        </S11>
    </Cnc>
</Report>
```

Logs S14

```
[2025/06/30 19:00:00.769] TASKS: Activated task task=9 (INFO)
[2025/06/30 19:00:00.775] TASKS: Error creating report task=9 request=S14
error="no nodes found" (WARN)
```

Report G01

```
<Report xmlns="http://stgdc/ws/G01" IdRpt="G01" IdPet="0" Version="4.0">
    <Cnc Id="CIR4622301207">
        <G01 Fh="20250704020000000S" Amed="9" Amax="9" Tot="9"
Aperc="100.00"></G01>
        <G01 Fh="20250704030000000S" Amed="9" Amax="9" Tot="9"
Aperc="100.00"></G01>
        <G01 Fh="20250704040000000S" Amed="9" Amax="9" Tot="9"
Aperc="100.00"></G01>
        <G01 Fh="20250704050000000S" Amed="9" Amax="9" Tot="9"
Aperc="100.00"></G01>
        <G01 Fh="20250704060000000S" Amed="9" Amax="9" Tot="9"
Aperc="100.00"></G01>
        <G01 Fh="20250704070000000S" Amed="9" Amax="9" Tot="9"
Aperc="100.00"></G01>
        <G01 Fh="20250704080000000S" Amed="9" Amax="9" Tot="9"
Aperc="100.00"></G01>
        <G01 Fh="20250704090000000S" Amed="9" Amax="9" Tot="9"
Aperc="100.00"></G01>
        <G01 Fh="20250704100000000S" Amed="9" Amax="9" Tot="9"
Aperc="100.00"></G01>
        <G01 Fh="20250704110000000S" Amed="9" Amax="9" Tot="9"
Aperc="100.00"></G01>
        <G01 Fh="20250704120000000S" Amed="9" Amax="9" Tot="9"
Aperc="100.00"></G01>
        <G01 Fh="20250704130000000S" Amed="9" Amax="9" Tot="9"
Aperc="100.00"></G01>
        <G01 Fh="20250704140000000S" Amed="9" Amax="9" Tot="9"
Aperc="100.00"></G01>
        <G01 Fh="20250704150000000S" Amed="9" Amax="9" Tot="9"
Aperc="100.00"></G01>
        <G01 Fh="20250704160000000S" Amed="9" Amax="9" Tot="9"
Aperc="100.00"></G01>
        <G01 Fh="20250704170000000S" Amed="9" Amax="9" Tot="9"
Aperc="100.00"></G01>
        <G01 Fh="20250704180000000S" Amed="9" Amax="9" Tot="9"
Aperc="100.00"></G01>
        <G01 Fh="20250704190000000S" Amed="9" Amax="9" Tot="9"
Aperc="100.00"></G01>
        <G01 Fh="20250704200000000S" Amed="9" Amax="9" Tot="9"
Aperc="100.00"></G01>
        <G01 Fh="20250704210000000S" Amed="9" Amax="9" Tot="9"
Aperc="100.00"></G01>
        <G01 Fh="20250704220000000S" Amed="9" Amax="9" Tot="9"
Aperc="100.00"></G01>
        <G01 Fh="20250704230000000S" Amed="9" Amax="9" Tot="9"
Aperc="100.00"></G01>
        <G01 Fh="20250705000000000S" Amed="9" Amax="9" Tot="9"
Aperc="100.00"></G01>
        <G01 Fh="20250705010000000S" Amed="9" Amax="9" Tot="9"
Aperc="100.00"></G01>
```

```xml
    </Cnc>
</Report>
```

Report G02

```xml
<Report xmlns="http://stgdc/ws/G02" IdRpt="G02" IdPet="0" Version="4.0">
    <Cnc Id="CIR4622301207">
        <Cnt Id="AUX1023030017">
            <G02 Fh="20250705000000000S" Atime="1440" Nchanges="0" Aconc="1440"
Atimeperc="100.00" Ahourly="FFFFFF"></G02>
        </Cnt>
        <Cnt Id="ELS0032114924">
            <G02 Fh="20250705000000000S" Atime="1440" Nchanges="0" Aconc="1440"
Atimeperc="100.00" Ahourly="FFFFFF"></G02>
        </Cnt>
        <Cnt Id="ITE0141020666">
            <G02 Fh="20250705000000000S" Atime="1440" Nchanges="0" Aconc="1440"
Atimeperc="100.00" Ahourly="FFFFFF"></G02>
        </Cnt>
        <Cnt Id="ORB0000803879">
            <G02 Fh="20250705000000000S" Atime="1440" Nchanges="0" Aconc="1440"
Atimeperc="100.00" Ahourly="FFFFFF"></G02>
        </Cnt>
        <Cnt Id="SAG0125322720">
            <G02 Fh="20250705000000000S" Atime="1440" Nchanges="0" Aconc="1440"
Atimeperc="100.00" Ahourly="FFFFFF"></G02>
        </Cnt>
        <Cnt Id="SAG0186301287">
            <G02 Fh="20250705000000000S" Atime="1440" Nchanges="0" Aconc="1440"
Atimeperc="100.00" Ahourly="FFFFFF"></G02>
        </Cnt>
        <Cnt Id="SOG0000502076">
            <G02 Fh="20250705000000000S" Atime="1440" Nchanges="0" Aconc="1440"
Atimeperc="100.00" Ahourly="FFFFFF"></G02>
        </Cnt>
        <Cnt Id="ZIV0039808892">
            <G02 Fh="20250705000000000S" Atime="1440" Nchanges="0" Aconc="1440"
Atimeperc="100.00" Ahourly="FFFFFF"></G02>
        </Cnt>
        <Cnt Id="ZIV0044603022">
            <G02 Fh="20250705000000000S" Atime="1440" Nchanges="0" Aconc="1440"
Atimeperc="100.00" Ahourly="FFFFFF"></G02>
        </Cnt>
    </Cnc>
</Report>
```

Report G12

```xml
<Report xmlns="http://stgdc/ws/G12" IdRpt="G12" IdPet="0" Version="4.0">
    <Cnc Id="CIR4622301207">
```

```
        <G12 Fh="20250704020000000S" Cpu="2" Ram="7" Flash="7" PerStor="7"
TmpStor="0" EthRx="0" EthTx="0" Plc="0" SerRx="0" SerTx="0" Temp="23"></G12>
        <G12 Fh="20250704030000000S" Cpu="2" Ram="7" Flash="7" PerStor="7"
TmpStor="0" EthRx="0" EthTx="0" Plc="0" SerRx="0" SerTx="0" Temp="23"></G12>
        <G12 Fh="20250704040000000S" Cpu="2" Ram="7" Flash="7" PerStor="7"
TmpStor="0" EthRx="0" EthTx="0" Plc="0" SerRx="0" SerTx="0" Temp="23"></G12>
        <G12 Fh="20250704050000000S" Cpu="2" Ram="7" Flash="7" PerStor="7"
TmpStor="0" EthRx="0" EthTx="0" Plc="0" SerRx="0" SerTx="0" Temp="23"></G12>
        <G12 Fh="20250704060000000S" Cpu="2" Ram="7" Flash="7" PerStor="7"
TmpStor="0" EthRx="0" EthTx="0" Plc="0" SerRx="0" SerTx="0" Temp="23"></G12>
        <G12 Fh="20250704070000000S" Cpu="2" Ram="7" Flash="7" PerStor="7"
TmpStor="0" EthRx="0" EthTx="0" Plc="0" SerRx="0" SerTx="0" Temp="23"></G12>
        <G12 Fh="20250704080000000S" Cpu="2" Ram="7" Flash="7" PerStor="7"
TmpStor="0" EthRx="0" EthTx="0" Plc="0" SerRx="0" SerTx="0" Temp="23"></G12>
        <G12 Fh="20250704090000000S" Cpu="2" Ram="7" Flash="7" PerStor="7"
TmpStor="0" EthRx="0" EthTx="0" Plc="0" SerRx="0" SerTx="0" Temp="23"></G12>
        <G12 Fh="20250704100000000S" Cpu="2" Ram="7" Flash="7" PerStor="7"
TmpStor="0" EthRx="0" EthTx="0" Plc="0" SerRx="0" SerTx="0" Temp="23"></G12>
        <G12 Fh="20250704110000000S" Cpu="2" Ram="7" Flash="7" PerStor="7"
TmpStor="0" EthRx="0" EthTx="0" Plc="0" SerRx="0" SerTx="0" Temp="23"></G12>
        <G12 Fh="20250704120000000S" Cpu="2" Ram="7" Flash="7" PerStor="7"
TmpStor="0" EthRx="0" EthTx="0" Plc="0" SerRx="0" SerTx="0" Temp="23"></G12>
        <G12 Fh="20250704130000000S" Cpu="2" Ram="7" Flash="7" PerStor="7"
TmpStor="0" EthRx="0" EthTx="0" Plc="0" SerRx="0" SerTx="0" Temp="23"></G12>
        <G12 Fh="20250704140000000S" Cpu="2" Ram="7" Flash="7" PerStor="7"
TmpStor="0" EthRx="0" EthTx="0" Plc="0" SerRx="0" SerTx="0" Temp="23"></G12>
        <G12 Fh="20250704150000000S" Cpu="2" Ram="7" Flash="7" PerStor="7"
TmpStor="0" EthRx="0" EthTx="0" Plc="0" SerRx="0" SerTx="0" Temp="23"></G12>
        <G12 Fh="20250704160000000S" Cpu="2" Ram="7" Flash="7" PerStor="7"
TmpStor="0" EthRx="0" EthTx="0" Plc="0" SerRx="0" SerTx="0" Temp="23"></G12>
        <G12 Fh="20250704170000000S" Cpu="2" Ram="7" Flash="7" PerStor="7"
TmpStor="0" EthRx="0" EthTx="0" Plc="0" SerRx="0" SerTx="0" Temp="23"></G12>
        <G12 Fh="20250704180000000S" Cpu="2" Ram="7" Flash="7" PerStor="7"
TmpStor="0" EthRx="0" EthTx="0" Plc="0" SerRx="0" SerTx="0" Temp="23"></G12>
        <G12 Fh="20250704190000000S" Cpu="2" Ram="7" Flash="7" PerStor="7"
TmpStor="0" EthRx="0" EthTx="0" Plc="0" SerRx="0" SerTx="0" Temp="23"></G12>
        <G12 Fh="20250704200000000S" Cpu="2" Ram="7" Flash="7" PerStor="7"
TmpStor="0" EthRx="0" EthTx="0" Plc="0" SerRx="0" SerTx="0" Temp="23"></G12>
        <G12 Fh="20250704210000000S" Cpu="2" Ram="7" Flash="7" PerStor="7"
TmpStor="0" EthRx="0" EthTx="0" Plc="0" SerRx="0" SerTx="0" Temp="23"></G12>
        <G12 Fh="20250704220000000S" Cpu="2" Ram="7" Flash="7" PerStor="7"
TmpStor="0" EthRx="0" EthTx="0" Plc="0" SerRx="0" SerTx="0" Temp="23"></G12>
        <G12 Fh="20250704230000000S" Cpu="2" Ram="7" Flash="7" PerStor="7"
TmpStor="0" EthRx="0" EthTx="0" Plc="0" SerRx="0" SerTx="0" Temp="23"></G12>
        <G12 Fh="20250705000000000S" Cpu="2" Ram="7" Flash="7" PerStor="7"
TmpStor="0" EthRx="0" EthTx="0" Plc="0" SerRx="0" SerTx="0" Temp="23"></G12>
        <G12 Fh="20250705010000000S" Cpu="2" Ram="7" Flash="7" PerStor="7"
TmpStor="0" EthRx="0" EthTx="0" Plc="0" SerRx="0" SerTx="0" Temp="23"></G12>
    </Cnc>
</Report>
```

Logs G01, G02 and G12

```
[2025/07/01 01:30:00.770] TASKS: Activated task task=10 (INFO)
[2025/07/01 01:30:02.784] TASKS: Finished request task=10 request=G01 (INFO)
[2025/07/01 01:30:02.800] TASKS: Finished request task=10 request=G02 (INFO)
[2025/07/01 01:30:02.814] TASKS: Finished request task=10 request=G12 (INFO)
[2025/07/01 01:30:02.822] TASKS: Finished task task=10 (INFO)
[2025/07/01 01:30:25.769] TASKS: Failed request in device request=T07
device=ZIV0044603022 retry=8 error="node ZIV0044603022 is in active key pending
failure" (WARN)
[2025/07/01 01:30:25.770] TASKS: Finished retry round request=T07 retry=8 (INFO)
[2025/07/01 01:40:34.770] TASKS: Failed request in device request=S04
device=ZIV0044603022 retry=2 error="node ZIV0044603022 is in active key pending
failure" (WARN)
[2025/07/01 01:40:34.770] TASKS: Finished retry round request=S04 retry=2 (INFO)
[2025/07/01 01:41:23.769] TASKS: Failed request in device request=S02
device=ZIV0044603022 retry=4 error="node ZIV0044603022 is in active key pending
failure" (WARN)
[2025/07/01 01:41:23.770] TASKS: Finished retry round request=S02 retry=4 (INFO)
[2025/07/01 01:41:51.769] TASKS: Failed request in device request=S05
device=ZIV0044603022 retry=4 error="node ZIV0044603022 is in active key pending
failure" (WARN)
[2025/07/01 01:41:51.769] TASKS: Finished retry round request=S05 retry=4 (INFO)
[2025/07/01 02:00:26.769] TASKS: Failed request in device request=T07
device=ZIV0044603022 retry=9 error="node ZIV0044603022 is in active key pending
failure" (WARN)
[2025/07/01 02:00:26.769] TASKS: Finished retry round request=T07 retry=9 (INFO)
[2025/07/01 02:10:00.769] TASKS: Data sent due TimeSendReq task=3 request=S04
(INFO)
[2025/07/01 02:10:35.769] TASKS: Failed request in device request=S04
device=ZIV0044603022 retry=3 error="node ZIV0044603022 is in active key pending
failure" (WARN)
[2025/07/01 02:10:35.769] TASKS: Finished retry round request=S04 retry=3 (INFO)
[2025/07/01 02:11:24.769] TASKS: Failed request in device request=S02
device=ZIV0044603022 retry=5 error="node ZIV0044603022 is in active key pending
failure" (WARN)
[2025/07/01 02:11:24.769] TASKS: Finished retry round request=S02 retry=5 (INFO)
[2025/07/01 02:11:52.769] TASKS: Failed request in device request=S05
device=ZIV0044603022 retry=5 error="node ZIV0044603022 is in active key pending
failure" (WARN)
[2025/07/01 02:11:52.769] TASKS: Finished retry round request=S05 retry=5 (INFO)
[2025/07/01 02:30:27.769] TASKS: Failed request in device request=T07
device=ZIV0044603022 retry=10 error="node ZIV0044603022 is in active key pending
failure" (WARN)
[2025/07/01 02:30:27.769] TASKS: Finished retry round request=T07 retry=10 (INFO)
[2025/07/01 02:40:36.769] TASKS: Failed request in device request=S04
device=ZIV0044603022 retry=4 error="node ZIV0044603022 is in active key pending
failure" (WARN)
[2025/07/01 02:40:36.769] TASKS: Finished retry round request=S04 retry=4 (INFO)
[2025/07/01 02:41:24.769] TASKS: Failed request in device request=S02
device=ZIV0044603022 retry=6 error="node ZIV0044603022 is in active key pending
failure" (WARN)
```

```
[2025/07/01 02:41:24.770] TASKS: Finished retry round request=S02 retry=6 (INFO)
[2025/07/01 02:41:53.769] TASKS: Failed request in device request=S05
device=ZIV0044603022 retry=6 error="node ZIV0044603022 is in active key pending
failure" (WARN)
[2025/07/01 02:41:53.769] TASKS: Finished retry round request=S05 retry=6 (INFO)
```

9.3 *APPENDIX III: VOLTAGE REGULATION ALGORITHM*

```
# alias python=/opt/venv/bin/python # to use the correct virtual environment
import requests
import html
import xml.etree.ElementTree as ET
from datetime import datetime
from influxdb_client import InfluxDBClient, Point, WritePrecision
from influxdb_client.client.write_api import SYNCHRONOUS
import time
import math

########## Function Definition #######################
# --- Function to calculate tap adjustment ---

def calculate_tap(meter_voltages, nominal=230, tolerance=3):
    # nominal = expected nominal voltage
    # tolerance = % of maximum and minimum variation from nominal voltage
    values = []
    for voltages in meter_voltages.values():
        values.extend([voltages['L1v'], voltages['L2v'], voltages['L3v']])

    if not values:
        return None

    average_voltage = sum(values) / len(values)
    print("Average Voltage: ", average_voltage, "V")
    lower_bound = nominal * (1 - tolerance / 100)
    upper_bound = nominal * (1 + tolerance / 100)

    if average_voltage < lower_bound:
        return 1
    elif average_voltage > upper_bound:
        return -1
    else:
        return 0

###################### End of Function Definition #####################

# Initial Data
tap_state = 0  # Initial tap state assumed to be 0
min_tap = -8   # Minimum tap position (depends on transformer manufacturer)
```

```
max_tap = 8     # Maximum tap position

# --- PART 1: SOAP request to the data concentrator ---

IP_DC = "cnc-service"
ID_PET = 10
ID_RPT = "S21"
ID_METERS =
"ZIV0047922435,SAG0166016473,SAG0166016496,ZIV0041674575,ZIV0047545478"
PRIORITY = 0
SOURCE = "DCC"
REQUEST_TYPE = "Request"

INFLUX_URL = "  # Make sure to set the correct URL"
INFLUX_TOKEN = "bbrtoken"
INFLUX_ORG = "bbrorg"
INFLUX_BUCKET = "bbrbucket"

client = InfluxDBClient(url=INFLUX_URL, token=INFLUX_TOKEN, org=INFLUX_ORG)
write_api = client.write_api(write_options=SYNCHRONOUS)

while True:
    print("Requesting data...")

    soap_body = f"""<?xml version="1.0" encoding="utf-8"?>
    <soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
      <soap:Body>
        <{REQUEST_TYPE} xmlns="urn:stgdc/dc/ws">
          <IdPet>{ID_PET}</IdPet>
          <IdRpt>{ID_RPT}</IdRpt>
          <Ffin></Ffin>
          <tfStart></tfStart>
          <tfEnd></tfEnd>
          <IdMeters>{ID_METERS}</IdMeters>
          <Priority>{PRIORITY}</Priority>
        </{REQUEST_TYPE}>
      </soap:Body>
    </soap:Envelope>"""

    headers = {
        "Content-Type": "text/xml; charset=utf-8",
        "SOAPAction": f"\"urn:stgdc/dc/ws/{REQUEST_TYPE}\""
    }

    try:
        response = requests.post(f"http://{IP_DC}:8080/WS_DC/WS_DC.asmx",
data=soap_body, headers=headers)
        print("Status code:", response.status_code)
        decoded = html.unescape(response.text)

        root = ET.fromstring(decoded)
        ns = {'s21': 'http://stgdc/ws/S21'}
        counters = root.findall('.//s21:Cnt', ns)
        influx_points = []
        meter_voltages = {}

        for cnt in counters:
            cnt_id = cnt.attrib.get('Id')
            err_cat = cnt.attrib.get('ErrCat')
            err_code = cnt.attrib.get('ErrCode')
            if err_cat == '2' and err_code in ['1', '2', '3']:
                continue
            s21 = cnt.find('s21:S21', ns)
            if s21 is None:
                continue

            timestamp_str = s21.attrib.get('Fh')
            timestamp = datetime.strptime(timestamp_str[:14], "%Y%m%d%H%M%S")

            def parse_float(attr): return float(s21.attrib.get(attr, 0))
            def parse_int(attr):
                val = s21.attrib.get(attr, "0")
                try:
                    return int(val)
                except (ValueError, TypeError):
                    return 0

            fields = {
                'L1v': parse_float('L1v'),
                'L2v': parse_float('L2v'),
                'L3v': parse_float('L3v'),
                'I3': parse_float('I3'),
                # 'Fc': parse_int('Fc'), # Phase in which the meter is located. 4
= All phases.
                'AIa': parse_int('AIa'),
                'AEa': parse_int('AEa'),
                'Eacti': parse_int('Eacti'),
                'Eanti': parse_int('Eanti'),
                # Active power imported/exported per phase (float)
                'Pimp1': parse_float('Pimp1'),
                'Pexp1': parse_float('Pexp1'),
                'Pimp2': parse_float('Pimp2'),
                'Pexp2': parse_float('Pexp2'),
                'Pimp3': parse_float('Pimp3'),
                'Pexp3': parse_float('Pexp3'),
                # Reactive power imported/exported per phase (float)
                'Qimp1': parse_float('Qimp1'),
                'Qexp1': parse_float('Qexp1'),
                'Qimp2': parse_float('Qimp2'),
                'Qexp2': parse_float('Qexp2'),
                'Qimp3': parse_float('Qimp3'),
                'Qexp3': parse_float('Qexp3'),
            }
            meter_voltages[cnt_id] = {
                'L1v': fields['L1v'],
                'L2v': fields['L2v'],
                'L3v': fields['L3v']
            }

            three_phase_voltage = (fields['L1v'] + fields['L2v'] + fields['L3v'])
/ 3 * math.sqrt(3)
            fields['ThreePhaseVoltage'] = three_phase_voltage

            pp = s21.attrib.get('PP')
            print("DEBUG: Phase Presence", pp)
            if pp:
                fields['PP'] = pp

            point = Point(cnt_id).time(None)
            # time(timestamp, WritePrecision.NS)
            for key, value in fields.items():
                point.field(key, value)

            influx_points.append(point)

        if influx_points:
            write_api.write(bucket=INFLUX_BUCKET, org=INFLUX_ORG,
record=influx_points)
            print(f"{len(influx_points)} points inserted into InfluxDB.")
        else:
            print("No valid data found to insert.")

    except Exception as e:
        print("Execution error:", e)

    print(f"[{datetime.utcnow()}] Executing OLTC...")

    nominal = 230 # V
    tolerance = 1 # %
    tap = calculate_tap(meter_voltages, nominal, tolerance)

    if tap is not None:
        # Create tap decision point
        tap_state += tap
        tap_state = max(min_tap, min(max_tap, tap_state))  # Clamp between min
and max
        tap_decision_point = Point("Tap").field("decision", tap).time(None)
        write_api.write(bucket=INFLUX_BUCKET, org=INFLUX_ORG,
record=tap_decision_point)
        tap_state_point = Point("Tap").field("State", tap_state).time(None)
        write_api.write(bucket=INFLUX_BUCKET, org=INFLUX_ORG,
record=tap_state_point)
        print(f"[{datetime.utcnow()}] Tap decision written: {tap}")
        print(f"[{datetime.utcnow()}] Tap state written: {tap_state_point}")
    else:
        print(f"[{datetime.utcnow()}] Insufficient data to make a decision.")
    print("Waiting 5 seconds...\n")
    time.sleep(5)
```

9.4 APPENDIX IV: SUSTAINABLE DEVELOPMENT GOALS

This project aligns with five key Sustainable Development Goals (SDGs), out of the seventeen defined by the United Nations Organization in 2015 [26]. By modernizing the low-voltage grid infrastructure increasing its efficiency through virtualization and edge computing this project aligns with the following goals:

- **Affordable and Clean Energy (Goal 7):** The deployment of containerized VDCs enhances the reliability, efficiency, and flexibility of the LV distribution network, indirectly optimizing energy delivery, supporting the integration of distributed resources and reducing the CAPEX, promoting the use of a single generic hardware instead of multiple specialize devices.
- **Industry, Innovation, Technology and Infrastructure (Goal 9):** The implementation and use of edge computing, containerization and standardized protocols in secondary substations favors the transition towards more resilient and scalable utility infrastructures, which helps the United Nations to build more reliable electrical infrastructures.
- **Sustainable Cities and Communities (Goal 11):** This project also contributes to the development of more sustainable urban mobility systems, allowing a more efficient integration of electric vehicles due to the introduction of edge computing in SSs that enhances monitoring and control of the LV network and enables the use of local smart charging algorithms that optimizes EV charging and prevents electrical congestion in urban areas.
- **Responsible consumption and production (Goal 12):** The transition from traditional hardware-based concentrators to software-based systems that is proposed in this project, aims to reduce the need of single-purpose devices, avoiding hardware obsolescence and promoting the reuse of general-purpose industrial computers. This solution reduces the

generation of electronic waste aligning with SDG 12's idea of sustainable resource use and circular production processes.

- **Climate Action (Goal 13):** The introduction of virtualization and edge computing in SSs also facilitates the integration of DERs, supporting higher penetration of EVs and therefore contributing to the reduction of CO2 emissions and aligning with the objective of SDG 13 of taking urgent action to combat climate change.